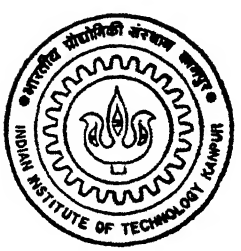
MULTICRITERIA ROBUST DESIGN USING NONDOMINATED SORTING GENETIC ALGORITHMS

by

Puneet Sharma



DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

March, 1996

MULTICRITERIA ROBUST DESIGN USING NONDOMINATED SORTING GENETIC ALGORITHMS

A Thesis Submitted

In Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

PUNEET SHARMA

to the

DEPARTMENT OF INDUSTRIAL & MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

MARCH, 1996

CERTIFICATE

This is to certify that the present work on " Multi-Criteria robust Design Using Nondominated Sorting Genetic Algorithms," by Puneet Sharma has been carried out under my supervision and has not been submitted elsewhere for award of a degree

(Tapan P. Bagchi)
Professor
Industrial & Management Engineering
Indian Institute of Technology
Kanpur- 208 016

22 March, 1996

25/3/96

ACKNOWLEDGMENTS

First and foremost, I express my heartfelt thanks to Professor Tapan P. Bagchi, whose guidance enabled me to venture into this topic. I am grateful for his valuable suggestions and excellent supervision throughout the span of this work.

I am thankful to my colleagues Mr Ajay Kansal and Mr. K. C. Jha for their invaluable help and support, without which my M Tech would have been probably incomplete I am grateful also to Professor K. Deb of Mechanical Engineering Dept., who explained NSGA to me and generously shared topics and references on GA with me. I further express my thanks to Mr. Manish Jain and Mr. Amarendra Kumar (M.Tech ME) for their suggestions and help during the course of my thesis work. Last but not the least, I extend my thanks to all the professors, my colleagues and the whole IME family who made my stay in IITK an enjoyable period

March, 1996

PUNEET SHARMA

ABSTRACT

In today's competitive world, quality of products is considered to be the most important requirement for an organization to survive and grow. Quality control practices such as control charts, SQC, SPC, Sampling and inspection etc which come in the category of *On-line quality control* are fast becoming obsolete or insufficient Today the emphasis is shifting to *Off-line quality control,* that is, to build quality into product or process at its *design* stage. This is done through robust design techniques. Taguchi proposed a method for it called Taguchi's "two-step method." However, this method has certain limitations, such as it cannot be applied to problems where individual factor effects interact or where two or more functional requirements (FRs) are to be made robust simultaneously. Bagchi and Kumar proposed an alternative method called "Constrained Optimization approach" for such situations. This method is used in the present work to a electronic filter design problem, where two FRs require that the design's cutoff frequency be equal to 6 84 Hz. and its galvanometer deflection be equal to 3.00 inches The design is made robust to noise due to tolerances in the component values. However, this method is also limited· it optimizes only one FR at one time. The present work shows that the use of NSGA in COP can produce *multiple Pareto-optimal solutions*, often a desired objective in engineering design The results thus produced are compared to solutions obtained by the earlier methods The solutions by NSGA appear to be somewhat more robust when compared with the earlier solutions while the NSGA solutions are also Pareto-optimal As in recent GA parameterization studies, we used the design of experiments method to find the optimum GA parameters for implementing the NSGA.

TABLE OF CONTENTS

	Page
LIST OF FIGURES	**vii**
LIST OF TABLES	**ix**
CHAPTER 1 THE ROBUST DESIGN METHODOLOGY	
1.1 Robust design and Taguchi methods	1
1.2 Robust design Methodology.	3
1.3 Orthogonal Arrays	6
1.4 Taguchi methods	7
1.5 Multi-objective decision making	10
1.6 Drawbacks of Classical methods	14
1.7 Unsolved problems in robust design.	15
CHAPTER 2 AN OVERVIEW OF GENETIC ALGORITHMS	
2.1 Introduction.	18
2 2 Features and structures of GA	19
2 3 Design issues of GA	19
2.4 SGA Simple Genetic algorithm	28
CHAPTER 3 THE PASSIVE FILTER DESIGN PROBLEM AND A STUDY OF ITS RESPONSES	
3.1 Statement of the problem.	32
3.2 The advantages of Constrained optimization.	41
3.3 The Constrained optimization robust design approach.	42
3.4 Application of Constrained optimization approach to Filter design problem.	44

3.5 Selection of optimum GA parameters 47

3.6 Experimental setup for selection of the best GA parameters. 49

3 7 The experimental results and their interpretation. 51

CHAPTER 4 PARETO-OPTIMAL SOLUTIONS USING MULTIOBJECTIVE GA

4 1 Nondominated Sorting Genetic algorithm 65

4.2 Sharing of dummy fitness among members on a front. 69

4.3 The Primary NSGA subroutines 71

4.4 The Concept of Sharing 72

CHAPTER 5 APPLICATION OF NSGA TO THE FILTER DESIGN PROBLEM. 79

CHAPTER 6 CONCLUSIONS AND SCOPE FOR FURTHER WORK 90

REFRENCES 92

LIST OF FIGURES

S. No.	Figure No.	Title of figure.	Page.
1	1.1	Taguchi design representation.	8
2	1.2	Parametric Experimental Plan.	9
3	1.3	Examples of interactions	11
4	1.5.1	Pareto-optimal front	12
5	1 4	Passive Filter Network	17
6	1 5	OTL Push-Pull Ckt.	17
7	2 1	SGA Flowchart.	31
8	3 1 1	Passive Filter Network	33
9	3 1.2	The Essence of COP	43
10	3.2 1-3.2.12	Interaction effects between design parameters.	55-58
11	3.3a	Variation of Variance of wc with R3.	59
	3.3b	Variation of Variance of D with R3.	59
12	3.4a-c	Factorial effects between parameters of GA	60
13	3.5a-c	Variation of Average, maximum and Minimum value of variance of wc with inner array experiments	61-62
14	3 6a-b	Variation of range of maximum and minimum of average variance of wc with inner array experiments	63
15	3.7	Convergence towards the best solution with generation for expe. # 20	64
16	4.1	NSGA Flowchart.	69
17	4.2	Comparison of GA with and without sharing for function f1	79
18	4.3	Comparison of GA with and without sharing for	

		function f2.	80
19	5 1	The Pareto-optimal front at generation 0.	84
20	5.2	The Pareto-optimal front at generation 10.	85
21	5.3	The Pareto-optimal front at generation 20.	86
22	5 4	The Pareto-optimal front at generation 30	87
23	5.5	The Pareto-optimal front at generation 40.	88
24	5 6	The Pareto-optimal front at generation 50.	89

LIST OF TABLES

S. No.	Table No.	Title of the Table.	Page
1	3 1	Treatment levels for three design parameters	34
2	3 2	Standard L9 OA	35
3	3.3	Combination of design parameter treatments in L9.	35
4	3.4	Levels for Noise factors	36
5	3 5	Standard L27 OA	37
6	3 6	Results of Outer array for row no 1 of inner array.	38
7	3.7	Average and Variance of wc and D obtained with outer array runs for all the nine rows of inner array.	39
8	3 8	The feasible range of R3.	45
9	3.9	Factor levels for SGA Parametric study.	50
10	3.10	Factor levels for SGA Parametric study expressed in terms of string length (l).	50
11	3.11	3X3X3 Experimental setup.	50
12	3.12	Minimum value of variance of wc obtained in 27 inner array experiments for five replicates	52
13	3 13	Average, Maximum and Minimum of the best solution in the five replicates	53
14	3.14	Convergence of variance towards best solution with generation.	64
15	4.1	Illustration of NSGA algorithm.	67
16	5.1	Number of solutions after each ten generations in NSGA run.	79

17	5.2	Solutions obtained by Bagchi and Kumar.	80
18	5.3	Statistics of the 25 solutions taken at random from generation # 50 in NSGA run.	81
19	5.4	Comparison between the solutions of the two methods	82

CHAPTER 1

THE ROBUST DESIGN METHODOLOGY

1.1 ROBUST DESIGN AND TAGUCHI METHODS

Competitiveness in international markets is now recognized as an essential element in national economic development Quality of products, as perceived by the customers, is emerging as the most important factor in facing this competition. In order to achieve competitive levels of quality, a company requires leadership from the top, good management practices, effective and efficient operating systems, and the use of appropriate *state-of -the -art* engineering design approaches. Engineering Design continuously plays an important role in manufacturing because of the shorter product lives observed by the users. Although statistical methods, in particular, experimental design, have existed and been applied to many disciplines for many years, classical manufacturing and engineering approaches were too often viewed as deterministic until recently when, the quality engineering approach proliferated because of Taguchi's effort in promoting experimental design [35].

The utmost goal for manufactured products should be to enhance human living standards, which, however is not achievable through cost-accounting principles. Instead it is quality that warrants profit because of world competitiveness and short product life cycle. Such a quality warranty can be provided, according to Taguchi, by using experimental design for products or processes, so that they are robust to environmental conditions and component variations and they minimize variation around a target values.

A design is called robust *if the product or process performs consistently on target in the field or in the hands of the user* and hence is insensitive to factors that are difficult to control [1]. Statistical quality-control techniques are widely used to improve the yield of manufacturing operations, the quality of the final product, and the reliability of the product in service. The best means of assuring good quality and highly

reliable products are through the proper design of products, manufacturing processes and systems. Axiomatic design enables us to approach the design process systematically and make correct design decisions. Proper design is one of the key elements for assuring high quality products because quality cannot be solely derived from shop floor. Rational design of products, manufacturing processes, and manufacturing systems have a direct bearing on the delivery of high quality products

Designing high quality product and processes at low cost is an economic and technological challenge to the engineer. A systematic and efficient way to meet this challenge is a new method of design optimization for performance, quality, and cost. The method called Robust Design consists of [1]

1. Making product performance insensitive to raw material variation, thus allowing the use of low grade material and components in most cases,

2. Making design robust against manufacturing variation, thus reducing labor and material cost for rework and scrap,

3. Making the designs least sensitive to the variation in operating environment, thus improving reliability and reducing operating cost, and

4 Using a new structured development process so that engineering time is used most productively.

. Before discussing Robust Design in detail we will try to understand " What is Quality."

The word *Quality* means different things to different people. Let us state what we mean by the ideal quality which can serve as a reference point for measuring the quality level of a product. *"The ideal quality a customer can expect is that every product delivers the target performance each time the product is used, under all intended operating conditions and throughout its intended life, with no harmful side effects."*[1]

When a product's performance deviates from the target performance, its quality is considered inferior. The performance may differ from one unit to another or from one environmental condition to another, or it might deteriorate before the expiration of

the intended life of the product. Such deviation in performance causes loss to the user of the product, the manufacturer of the product and in varying degree , to the rest of the society as well. Following Taguchi, we measure *the quality of a product in terms of the total loss to the society due to functional variation and harmful side effects.* Under the ideal quality the loss would be zero; the greater the loss, lower the quality.

The three main elements of costs are ***Operating cost, Manufacturing cost and R & D cost.*** The manufacturing cost and the R & D cost are incurred by the producer and passed to the customer. The operating cost is directly borne by the customer and it is directly related to the product quality Robust design is a systematic method for keeping the producer's cost low while delivering a high quality product, that is, while keeping the operating cost low [1]

The Fundamental Principle of Robust Design is to improve the quality of product by minimizing the effects of causes of variation without eliminating the causes, or more precisely this an strategy for minimization of total cost This is achieved by optimizing the product and process design to make the performance minimally sensitive to the various causes of variation This is called *parameter design.* However parameter design alone can not always lead to sufficiently high quality. Further improvement can be obtained by controlling the causes of variation where economically justifiable, typically by using more expensive equipment, higher grade, better environmental controls, etc , all of which lead to higher product cost, or operating cost, or both. The benefits of improved quality must justify the added product cost.

There are two important tasks to be performed in robust design :-

1.Measurement of Quality during Design/Development.

2.Efficient experimentation to find dependable information about the Design Parameters.

1.2 ROBUST DESIGN METHODOLOGY

Robust design methodology offers simultaneous improvement of product quality, performance, cost, and engineering productivity. It's wide spread use in

industry has a far reaching economic impact, because this methodology can be applied in all engineering activities, including product design and manufacturing process design The following are the key concepts in connection with robust design methodology -

1.2.1 Quality Loss Function and Fraction defect fallacy:-Previously it used to be the common practice to measure quality in terms of *fraction defective,* which is often incomplete and misleading It implies that product which meet the specifications (allowable deviation from the target response) are equally good and those which are outside the specification limit are bad. The fallacy here is that the product that barely meets specifications, from customer's point of view is as good as that is barely outside the specifications [1]

Taguchi gave a new definition to quality, according to him: " *A product has the ideal Quality when it delivers on target performance each team the it's user uses the product, under all intended operating conditions throughout its intended life and society (either manufacturer or the customer) incurs a loss if the product fails to perform on target".* He gave a measure of the quality called ***quadratic loss function.*** According to this function the quality loss is given as

$$L(y) = Ky^2$$

There are number of variations to this function according to the application [1]:-

* **Smaller the better type:-** Such as in case of radiation leakage from microwave oven,

$$L(y) = Ky^2$$

* **Larger the better type:-** Such as the bond strength of a adhesive which can not be negative but zero is their worst value,

$$L(y) = k[\ 1/y^2\ ]$$

* **Nominal the best type :-** When the product has to meet some desired target value

$$L(y) = k\ (y-m)^2$$

*** Asymmetric lose function:-** In certain situations the deviation of target function has a more harmful effect in one direction than other,

$$L(y) = | k1 (y-m)^2 \text{ if } y > m,$$

$$= | k2 (y-m)^2 \text{ if } y \leq m.$$

The value of k can be determined using functional limits of y[1].

1.2.2 CAUSES OF VARIATION (NOISE FACTORS):- The following are the causes of variations:-

1. External causes :- The environment in which a product works and the load to which it is subjected to are the two main external sources of variation of a product's function.

2. Unit to Unit variation :- The variation that is inevitable in a manufacturing process leads to variation in the product parameters from unit to unit.

3. Deterioration :- As the time passes the values of individual components may change leading to the deterioration in the product performance.

Average Quality Loss ;- If m is the target, y be the nominal the best type quality characteristic. Then the average quality loss Q will be; [1]

$$Q = k[L(y1) + L(y2) \text{ ----------} + L(yn)]$$

$$= k [\sum_{i=1}^{n} (y1 - m)^2)]$$

$$= k[(\mu - m)^2 + (n-1)\sigma^2/n]$$

$$= k[(\mu - m)^2 + \sigma^2]$$

$$(\text{for } n >> 1)$$

Thus the average quality loss has following two components :

(i) $k(\mu - m)^2$ resulting from deviation from the average value of y from target,

(ii) $k\sigma^2$ resulting from mean squared deviation of y around its own mean.

It is easy to eliminate the first by adjustment. Reducing the second is more difficult. The three methods of reducing the second (variance) in order of cost effectiveness are:-

1. Screening out the bad products.

2. Discovering the causes of malfunction and eliminating it

3. Application of Robust Design Methodology.

The first two causes come in the category of *on-line quality control techniques,* whereas the third comes in the category of *off-line quality control*. This consists of making of a product's performance insensitive to noise factors. Usually, a product's quality characteristic is related to the product parameters and noise factors through a complicated, nonlinear function The principal goal of Robust Design is to exploit the non linearity and noiseXdesign parameter interactions to find a combination of product parameters values that give the smallest variation in quality characteristic around the desired target value[1]. Taguchi's design strategy has three steps:

1. Concept design:- In this step, the designer examines a variety of architecture's and technologies for achieving the desired function of the product and selects the most suitable one for the product

2. Parameter design:- In parameter design, we determine the best settings for the control factors that do not affect manufacturing cost, that is, the settings that minimizes quality loss. Thus, we must minimize the sensitivity of the function of the product or process to all the noise factors and also get the mean function on target

3. Tolerance design:- In tolerance design, a trade off is made between reduction in the quality loss due to performance variation and increase in manufacturing cost. Tolerance design should be performed only after sensitivity is minimized through parameter design

Robust Design and its associated methodologies focus on parameter design.

The quality control activities and associated stages of a product's life cycle are:-

1. Product design.

2. Process design

The Quality control activities at these two stages are called *off- line quality control.*

3. Manufacturing:- The quality control activities at this stage are called *on-line quality control.*

4. Customer Usage:- At the time of customer usage of product, the quality control activities involve *warranty and service.*

1.3 ORTHOGONAL ARRAYS

Orthogonal arrays are special experimental designs that require only a small number of experimental trials to help discover *main factor effects.* OAs are "fractional factorial" designs and symmetrical subset of all combinations of treatments in corresponding full factorial design [2]. In cases where the factor effects are additive, linear and separable, OAs allow us to study the main factor effects of several design parameters at once and efficiently. OAs use the concept of "Orthogonality" which implies that the entries in the arrays satisfy a special mathematical condition. Suppose we define the Yi, a weighted sum of nine experimental observations x1, x2, x3,.....,x9, as

$$Yi = \sum_{j=1}^{9} wij.xj$$

Such that the weighted factors{wij} satisfy the condition

$$\sum_{j=1}^{9} wij = 0$$

Then one calls Yi a *contrast.* One calls the two contrasts Y1 and Y2 orthogonal if the inner product of the vectors {w1j}and {w2j} is zero. Thus, Y1 and Y2 are orthogonal if,

$$\sum_{j=1}^{9} w1j.w2j = 0$$

Any two columns used in the OAs are mutually orthogonal. The orthogonal arrays selected according to the number of factors and the levels of each individual factor. The Rules for selecting OA are explained in the references.[1][2].

1.4 TAGUCHI METHODS

Taguchi aimed at making the design robust first, followed by an adjustment to put the performance at the desired target. His methods enable the designs to achieve (a) Minimum dispersion in performance about the target. (b) Minimum sensitivity to variations transmitted from components and (c) Minimum sensitivity to environmental noise. His methods prescribes that the S/N ratio be maximized, the bias be minimized and that the standard be minimized Taguchi methods characterizes the design in terms of four input factors and an output response. The response is the output of the design which is characterized by the functional requirements (FRs). Signal factors (M) are parameters that can be set to achieve a specific response (FRs) Control factors (z) are DPs with which the designer can control the output responses (Frs). Scaling factors (R) are special cases of control factors that can be altered to achieve a desired relationship between a signal factor and the output response. Noise factors (x) are those variables in the design that are uncontrollable or unpredictable and represent an uncertainty that the desired output will be achieved. [3]:-

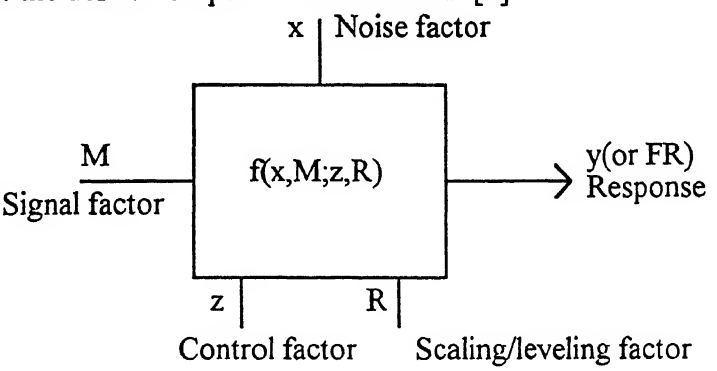


Fig 1.1 Taguchi design representation

A matrix developed by Taguchi (S/N) in order to optimize a design is the ratio of the mean in output response resulting from control factors to variation resulting from unpredictable or noise factors. The output response may be represented by a function $y = g(M, z, R) + e(x, M; z, R)$, Where g is predictable and e is unpredictable. To optimize y, the z and R can be chosen such that the predictability ratio is maximized. The S/N ratio N is expressed as the log10 of this predictability ratio, multiplied by 20 .

$$N(x , R) = 20 \log10 [g (z , R) / E (z, R)]$$

The S/N ratio is based on the analysis of variance technique that was introduced by Fisher and others Measurement are analyzed by the variance technique in order to determine the relative importance of the various effects. The S/N ratio is expressed as the ratio of mean and the variance Which may be written as.-

$$N = 20.\log10 E(\mu/ \sigma)$$

where μ = observed mean

σ = Standard deviation caused by noise.

The accuracy of this technique is dependent upon the sampled output response A sample of responses must be sufficiently large to represent fully the broad range of outcomes. Taguchi suggested the use of OAs for the experimentation The process of fitting an OA to a particular project has been made particularly possible by a graphical tool called ***Linear Graphs*** developed by Taguchi.[1][2].

1.4.1 INNER AND OUTER ARRAY:- Taguchi's robust design methodology uses an experimental design for the control factors Taguchi called this design matrix an *inner array.* For each individual row of *inner array,* we have to conduct a number of experiments taking the noise factors in the account, that is called the *outer array.* The simulation results of inner and outer arrays are analyzed, the appropriate statistics (mean , variance , S/N) are calculated , the main and adjustment parameters are identified using the technique of analyses of variance. The Figure below shows the Parameter Experimental plan used to observe the performance:- Fig 1.2

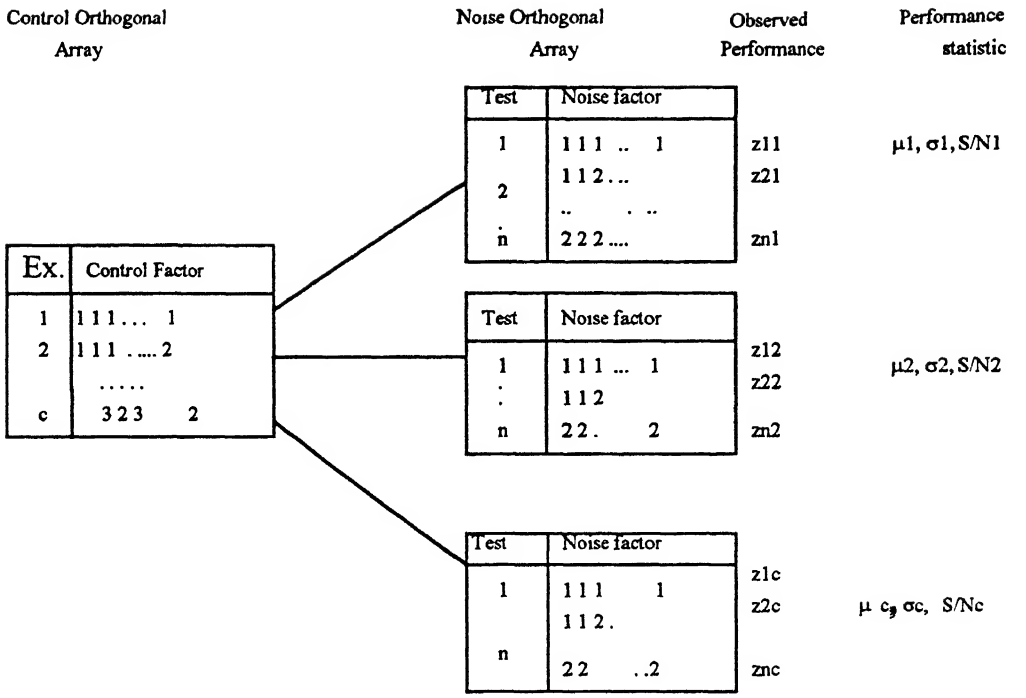


Fig 1.2 Parametric Experimental Plan.

1.4.2 TAGUCHI'S TWO STEP PROCEDURE TO ROBUST DESIGN

1. After conducting the orthogonally designed experiments, identify those control factors {z}(Fig 1.1) that show high signal to noise ratios (S/N). Set these factors at values that corresponds to the highest S/N ratios.

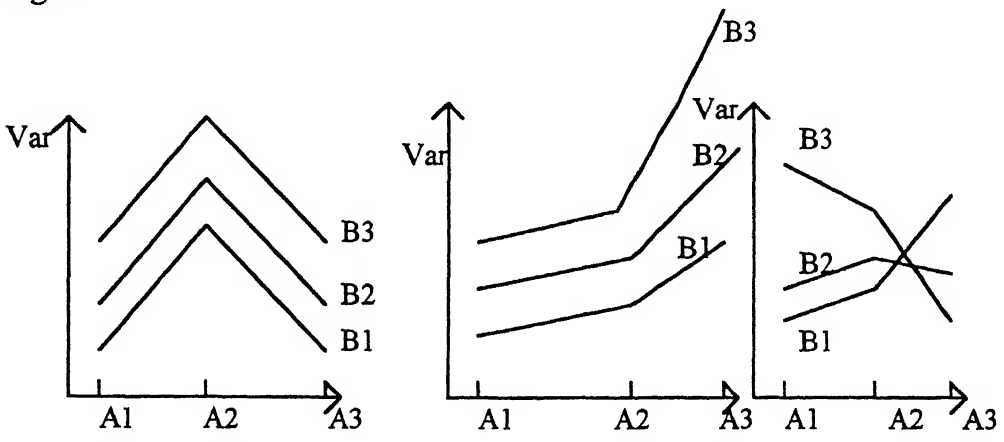
2 Identify the control factors that have no or little effect on S/N ratio but have the highest effect on *mean performance.* These control factors should be treated as the scaling factor (R). Adjust the value of R such that performance is on target.

Taguchi's two step method is based on the assumptions that the individual effects of the control factors are non interacting and they follow the principle of additivity. This permits the use of certain *partial factorial* designs. Additivity of effects leads to the major reduction in the number of experiments that needs to be run.

What if the individual factors are interacting? In such cases the number of experiments to be run are considerably high and Taguchi's two step method does not work effectively. The concept of interactions can be understood from figure 1.3 below The figure (a) shows the case of no interaction between the two factors A and B. Here

the lines of factor A for settings B1, B2 and B3 of factor B are parallel to each other. Parallel lines indicate that if we change the settings of factor A from A1 to A2 to A3 the corresponding change in Variance is same regardless of the settings of B. The Additive model is perfect in this case and Taguchi's two step method can be used in this situation In Figure (b) the lines are not parallel but the direction of improvement does not change In this case the optimum settings identified by the Additive model are still valid In Figure (c) not only the lines are not parallel but the direction of improvement is not consistent Taguchi's two step method can not be applied in this case. The purpose of verification experiment is to warn us when the additive model is not valid and prevent faulty product or process designs from going downstream.[1]

Fig-1.3



(a) No Interaction (b) Synergistic Interaction (c) Anti Synergistic Interaction.

Fig 1.3 Examples of Interactions

Presence of DPxDP interaction often induce nonlinearties in the underlying model and deemed to be highly undesirable by Taguchi. The presence of interactions makes the interpretation of main effects very difficult and often quite misleading. Valid inferences in such situations can be drawn only looking at the individual levels together with their interaction plots. This obviously becomes quite messy if the factors are at three or more levels. Moreover, if the DPxDP interaction are present, the optimum settings at the laboratory may not indeed be optimum under manufacturing or customer use conditions. Further difficulties with Taguchi methods arise when more than one responses are to be simultaneously brought on target and their variability

minimized. In such situations, it may be difficult to find adjustment factors (R) which can be manipulated to give on target performance without affecting S/N ratios.

1.5 MULTI CRITERIA DECISION MAKING

Multicriteria decision making (MCDM) has two major phases analytical and judgmental. While the analytical phase corresponds to the deterministic in nature and quantifiable aspect of decision making. The judgmental phase corresponds to the indeterministic and in nature, nonquantifiable aspect. The approaches to the MCDM are concerned with filling this " Phase gap" with rational procedure The method for treating the analytical phase of the Multi criteria decision process is called Multi criteria optimization. Multi criteria optimization has two key concepts [5].

1. Pareto optimality and
2. The preferred decision.

In general, decision with Pareto optimality are not uniquely determined, there are many Pareto optimal solutions. Due to this plurality of optimal decisions, the most desirable decisions should be selected from among the Pareto optimal solutions. The final solution as the most desirable or at least the best compromised, is called the preferred solution. The procedure that solves the Multi objective optimization is composed of three steps:

1. Mathematical modeling of the system structure,
2. Generation of the Pareto optimal solutions,
3. Selection of the preferred solution.

Steps (1) and (2) are embedded in process of solving the mathematical optimization problems. Step (3) should be inserted externally with subjective decisions All the solution sets generated in Pareto optimal front should have a simple and highly desirable property: nondominance.

A point within such a set is ***nondominated*** *if no other point is feasible at which the same or better performance could be achieved with respect to all criteria, with at least one being strictly better.* The nondominance concept originates in Pareto principle : A state of the world A is preferred to B if at least one person is better off in

A and nobody is worse off. To express nondominance in simple vector comparison, Let X and Y be the two vectors of n components, x1, x2, x3,----xn and y1, y2, y3,---yn respectively. Thus,

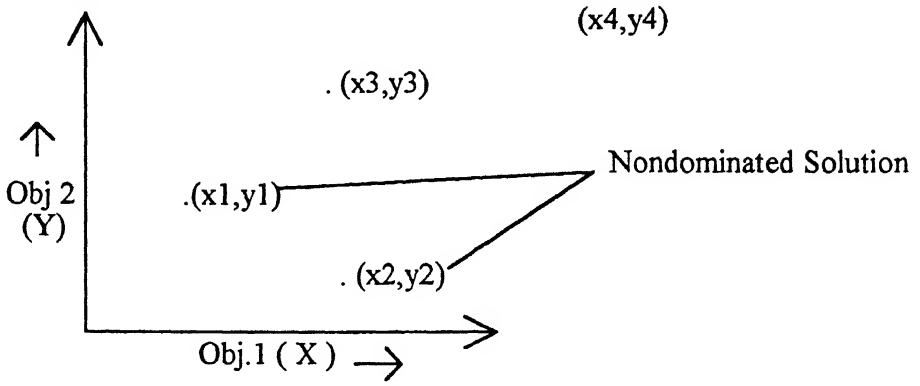


Fig 1.5.1 Nondominated (Pareto optimal) Solutions.

X = (x1, x2, x3..... .., xn) and Y = (y1, y2, y3 ....., yn)

We say that X dominates Y if, $x_i \geq y_i$; i = 1,2,3 ...,n and $x_i > y_i$ for at least one i The set of nondominated solutions is referred as Pareto-optimal set, admissible set, first front etc Let us assume that X and Y are two objectives to be minimized. The four solutions are (x1,y1), (x2,y2), (x3,y3) and (x4,y4) as shown in the Figure 1.5.1. Since the value of the two objective functions X and Y is higher at point (x3,y3) compared to (x1,y1) and (x2,y2). Therefore point (x3,y3) dominates points (x1,y1) and (x2,y2). Similarly (x4,y4) dominates points (x1,y1), (x2,y2) and (x3,y3). The front or the set of solutions consisting of (x1,y1) and (x2,y2) is called Pareto-optimal front or efficient front etc. and each solution in such a front is Pareto-optimal Solution There are various methods of solving Multi objective optimization problems some of which are discussed below:

1.5.1. Method of Objective Weighting:[4] In this method multiple objectives are combined in to one overall objective function, Z , as follows:

$$Z = \sum_{i=1}^{n} w_i . f_i(x) \ ,$$

where $x \rightarrow X$; f1(x),f2(x)---fn(x) are individual objective functions. X is a set of feasible solutions. The weights are fractional numbers ($0 \leq w_i \leq 1$) and all weights

summed up to 1 or $\sum_{i=1}^{N} wi = 1$. In this method the optimal solution is controlled by the weight vector wi The preference of one objective can be changed by changing the corresponding weight. Mathematically the solution obtained with equal weights to all objectives may offer least conflict In most real world cases each objective is first optimized and all function values are computed at each individual optimum solutions. There after depending on the importance of objectives a suitable weight factor is chosen and the single objective algorithm can be used for this problem The only advantage of this method is that the emphasis of one objective over the other can be controlled. The obtained solution is usually a Pareto optimal solution.

1.5.2. Method of Distance Function:- In this method the scalarization is achieved using a demand level vector y which has to be specified by the decision maker. The single objective function derived from multiple objective is as follows:

$$Z = \left[\sum_{i=1}^{N} \{fi(x) - yi\}^{r} \right]^{1/r} \qquad 1 \leq r < \infty$$

Here fi(x) are individual objective functions. Usually r = 2 is chosen, with yi as individual optimum of objectives. The solution obtained in this case depends on the chosen demand level vector. Arbitrary selection of a demand level vector will lead to a non Pareto optimal solution. The decision maker must have a thorough knowledge of each objectives prior to the selection of demand level vector. The difference in this method and the previous is that this method requires knowledge of goal for each objective function, whereas the previous method requires the relative importance of each objective function .

1.5.3 Min-Max Formulation:- This method attempts to minimize the relative deviation of the single objective function from the individual optimum. That it tries to minimize the objective conflict. For the minimization problem the corresponding min-max problem is formulated as follows:

$$\text{minimize } F(x) = \max \{Zj(x)\}, \quad j = 1,2,.....,N$$

where x ε X the (feasible region) and Zj(x) is calculated for non negative target optimal value $\underline{f_j} > 0$ as follows:

$$Z_j(x) = (f_j - \underline{f_j}) / \underline{f_j}, \; j = 1,2,3,---,N$$

This method can yield the best possible compromise solution when objectives with equal priority are required to be optimized

1.6 Drawbacks of Classical Methods :- In all the above discussed methods the multiple objective are combined to form one objective by using some knowledge of the problem being solved The optimization of the single objective may guarantee a Pareto optimal solution, but results in a single point solution. In real world the implementation of the single solution may not be practical or if some of the objectives are noisy and have discontinuous variable space there methods may not be effective The most profound drawback of these algorithms is their sensitivity towards weights or demand levels Thus for different situations different weight factor need to be used and the same problem needs to be solved a number of times

Keeping these drawbacks in mind a more realistic method would be one that gives multiple Pareto optimal [4][5] solutions simultaneously, so that decision makers may be able to choose the appropriate solution for a particular situation. Multiple Pareto-optimal solutions are also suitable when the previous situation has changed and a new solution is required to be implemented. Since the GAs deal with a population of points , multiple Pareto-optimal solutions can be captured in the population in the single run.

1.7 APPLICATIONS AND UNSOLVED PROBLEMS IN ROBUST DESIGN

The following are some of the problems that have been solved using the robust design methodology at AT&T and elsewhere.[1]

1.7.1 The ***window photolithography*** application was the first application in the United States that demonstrated the power of Taguchi's approach to quality and cost improvement through robust process design. The benefits of the application were:-

(i). Four fold reduction in the process variance.

(ii). 3 fold reduction in fatal defects.

(iii). 2 fold reduction in the processing time.

(iv). Easy adaptation of process to finer-line technology

1.7.2 The ***aluminum etching*** application originated from a belief that poor photoresist print quality leads to line width loss and to undercutting during the etching process. By making the etching process insensitive to photoresist profile variation, the visual defects were reduced from 80 percent to 15 percent

1.7.3 The ***reactive ion etching*** of tantalum silicide used to give highly non uniform etch quality, so only 12 out of 18 possible could be used for production After optimizing , 70 wafer positions became usable- a hefty 40 percent increase in the machine utilization and $ 1.2 million were saved by completing the project in 20 days deadline

1.7.4 In the **differential amplifier circuit optimization** application, an important quality characteristic is the offset voltage. Ideally the offset voltage should be zero. An application of design optimization technique resulted in a 40 percent reduction in the root mean square offset voltage by simply finding new optimum nominal values for the circuit parameters.

1.7.5 Passive filter design problem explained below is an example of a robust design problem with multiple performance characteristics. The problem is to design a passive electronic (Fig 1.4) filter such that the two performance characteristics namely the filter cut off frequency and the galvanometer deflection are on the their target values of 6.84 Hz and 3.00 inches and also the two response characteristics, namely , the variation of cut-off frequency and deflection around their target values [3][6][30]

1.7.6 The **Optimization of Strength of castings.** This is a design problem described by Tribus M. and Sozny G [31]. In which manufacturing process has to be made robust. The problem involves optimization of strength of castings produced by a screw molding process The problem involves six design parameters. The goal is to find a setting of the design DPs which would give the target casting strength of 160 units.

1.7.7 Improving a Printing Machines capability to apply coloring ink: This problem is discussed by Box and Draper [31]. The problem involves studying the

effects of speed(x1), pressure (x2), and distance (x3) upon a printing machine's ability to coloring ink upon package labels

1.7.8 Robust design of OTL push-pull circuit : In designing an output transformer less (OTL) push-pull circuit (Fig 1.5) an objective is to make the mid point voltage Vm stable around 6V in the presence of variation in the values of the components of the circuit (Internal noise). The mid-point voltage Vm is related to the other by the equation:

$$Vm = (Vb1 + Vb2)\beta Ro /(\beta Ro + Rf) + (Ec - Vbe3) Rf /(\beta Ro + Rf) + Vbe3\ \beta Ro\ Rf / (\beta Ro + Rf)Rc1$$

Where V = Rb2/ (Rb2 + Rb1), Ro = R0 + RL, RL = 9 ohm, Ec = 12 V

Vbe1 = Vbe3 = 0.65 V, Vbe2 = 0.72V, Rb1, Rb2, Rf, Rc1 and Rc2 are the resistances and β is the current gain

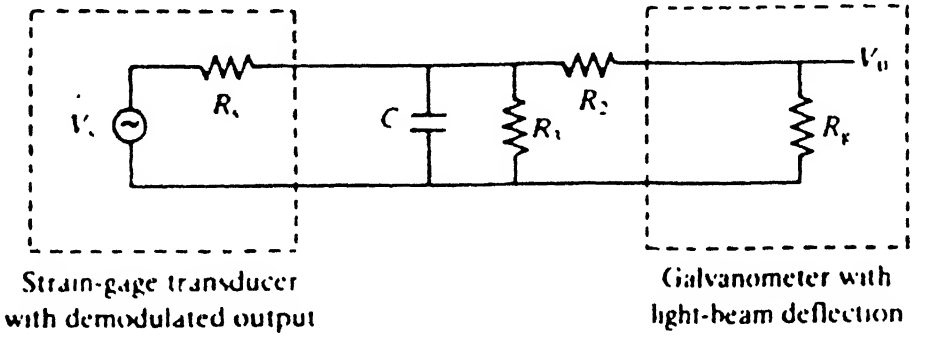


Figure 1.4 The RC passive filter network interfacing a strain-gauge transducer to a Galvanometer with a light beam deflection for the measurement of deflection.

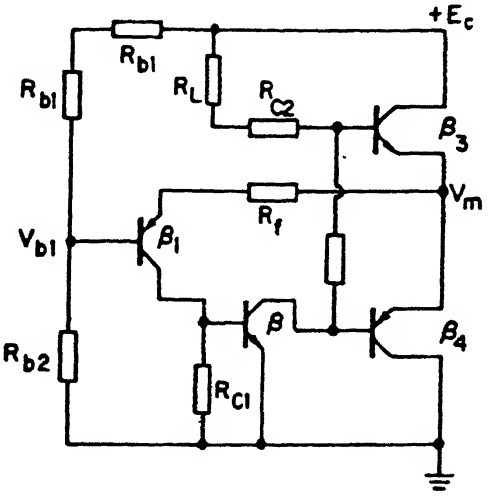


Fig 1.5 OTL Push-Pull Ckt

CHAPTER 2

AN OVERVIEW OF GENETIC ALGORITHMS

2.1 INTRODUCTION:- Genetic algorithms (GAs) have been developed by John Holland in the mid sixties at the University of Michigan. Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine the survival of the fittest among string structures to form a search algorithm with some of the innovative flair of human search. In every generation a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure They efficiently exploit historical information to speculate on new search points with expected improved performance. At each generation the relatively "good" solutions reproduce, while the relatively "bad" ones die. The following are the terms in connection with genetic algorithms (GA).

2.1.1 Population and Individual :-A Population is a set of potential solutions at hand while an individual is one such solution. The individuals are also called strings or chromosomes. A Chromosome is a coded representation of an individual solution

2.1.2 Genes and alleles :- The Chromosomes are made up of units arranged in linear succession (strings). Each individual unit is called a gene A gene is said to be in several states, called *alleles*. For coding using binary strings which is most frequently used, although other options can also be pursued, a gene can take only two values, 0 or 1. Here 0 or 1 are alleles of the gene.

Each chromosomes can be inverse mapped or decoded to its corresponding xi (real number) value. Putting these values in f(x), we can evaluate the objective function value. The higher the objective function value more the *fitness* of the string. The Utility function is what the user is interested in optimizing, while objective function is some what suitable transformation applied on the utility function for use in

some optimization algorithm The fitness of the string is always given by the objective function value.

2.2 FEATURES AND STRUCTURE OF GA

2.2.1 Features:- Genetic algorithm are different from traditional optimization and search procedure in the following four ways

1. GAs work with coding of parameters, not the parameters themselves

2. GAs search the optimal point from a population of points, not a single point.

3. GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.

4. GAs use probabilistic transition rules not deterministic rules.

2.2.2 Structure:- Here is the structure of the general genetic algorithm [6]. Let at time t (generation, iteration), P(t) denote the population.

```
procedure Genetic algorithm ;
begin
        t <- 0 ;
        initialize population P(t) ;
        evaluate population P(t) ;
        while ( not termination criteria ) do
        begin
             t <- t + 1 ;
             reproduction ;
             crossover ,
             mutation ,
             evaluate population P ( t + 1) ;
         end
  end ;
```

2.3 DESIGN ISSUES OF GA:- [6]

The major design issues are:

1 Representation,

2 Initialization,

3 Evaluation,

4 Parent selection,

5 Recombination operator,

6 Parameter values,

7 Replacement strategy,

8 Convergence policy

2.3.1 Representation

To solve any optimization problem using genetic algorithm, a coding scheme is necessary to encode the permutation of the problem into a string The scheme is problem dependent and not unique

2.3.2 Initialization

The next important step in GA is to generate the initial population, that is, create an initial population Here an initial population of desired size has been considered, on the entire population at each generation we apply the genetic operators. To implement this we utilize two non overlapping populations, the new offspring are created using the genetic operators on the members of *oldpop*, these new individuals are placed in *newpop*. There are more efficient methods of handling populations. We could maintain a single overlapping population and pay more attention to who replaces whom in successive generations There is also no particular reason to keep the population size constant

2.3.3 Evaluation

There are number of characteristics of the evaluation function that can enhance or hinder a GA's performance. Since the performance of GA is highly sensitive to the objective function, it is very important to scale the fitness values so as to diversify the population Further Goldberg[7] suggested that fitness scaling helps in differentiating the chromosomes with average fitness value and best fitness value. The fitness function described below gives even scaling effect over a large cost range [8].

$$y = (x - \log_e(x)) \times (10^k)$$

x = objective function value,

y = fitness value,

k = constant depends on the range of the expected total cost, and the rates of changes in the fitness values at different ranges of the expected cost.

The second important property corresponds to mapping the natural objective function to a fitness function. For minimization problem the following transformation can be applied [7]

$$f(x) = Cmax - g(x) \quad \text{When } g(x) > Cmax$$
$$= 0 \quad \text{otherwise}$$

g(x) is the objective function value and Cmax may be taken as an input coefficient, as the largest g value observed in the last k generation, or as the largest g value in the current population More appropriately , Cmax should vary depending upon the population variance.

Another transformation may be [7],

$$f(x) = 1 - (Cx - Cmin) / (Cmax - Cmin),$$

Where Cx - cost of solution x,

Cmin - cost of cheapest individual of population,

Cmax - cost of most expensive individual of population.

This gives a fitness value varying linearly , with cost from 0 for most expensive solution, to 1 for cheapest By amending the fitness function, it is possible to balance between a deterministic and a random method of selection A fitness value that is highly sensitive to cost will produce population of solution with little diversity, giving rapid convergence, possibly to a local optimum. On the other hand, a fitness value insensitive to cost gives a random search. An efficient GA will steer a middle course between these extremes.

2.3.4 Parent Selection

The purpose of parent selection in GA is to give more reproductive chances, on the whole to those population members that are most fit and form a mating pool. There exist a variety of operators in GA literature, but all of them share the same purpose of picking up good strings from the current population and inserting the

duplicates of them in the mating pool. Here we discuss some of the commonly used reproduction schemes

(i) Proportionate Selection :[8]

Steps·

1. Calculate the fitness value eval(Vi) for each chromosome Vi (for i =1,.., pop_size).

2. Sum the fitness of all the population members; call the result total fitness.

$$F = \sum\{ \text{eval}(Vi) \}$$

3. Calculate the probability of a selection pi for each chromosome Vi (for i = 1,...., pop_size).

$$pi = \text{eval}(Vi) / F$$

4. Calculate a cumulative probability qi for each chromosome Vi (for i = 1,...., pop_size).

$$qi = \sum pj$$

5. Generate n, a random number between 0 and total fitness value(1).

6. Return the first population member whose fitness, added to the fitness of the preceding members is greater than or equal to n

If $r \leq q1$ then select the first chromosome V1, other wise select the i^{th} chromosome Vi ($2 \leq i \leq pop\_size$) such that $qi-1 \leq r \leq qi$.

Example :

Chromosome	1	2	3	4	5	6	7	8	9	10
Fitness	8	2	17	7	2	12	11	7	3	7
Running Total	8	10	27	34	36	48	59	66	69	76
Random number	23	49	76	13	1	27	57			
Chromosome chosen	3	7	10	3	1	3	7			

The effect of roulette wheel selection is to return a randomly selected parent. Although the selection is random, each parent's chance of being selected is directly proportional to it's fitness. On balance, over a number of generation the algorithm will

drive out the least fit member and contribute to the spread of the genetic material in the fittest population member. Of course it is possible that the worst population member could be selected by the algorithm each time it is used Such an occurrence would inhibit the performance of a GA using this selection technique, but the odds of this happening in a population of any size are negligible. There have been many variations to the simple procedure proposed above. The first variation, named elitist model, enforces preserving the best chromosome in each generation. The second variation, the expected value model, reduces the stochastic errors of the selection routine This is done by introducing a count for each chromosome v, which is set initially to the f(v)/ f value and decreased by 0 5 or 1 when the chromosome is selected for reproduction with crossover and mutation, respectively. When the chromosome count falls below zero the chromosome is not available for selection any longer [9].

(ii) Tournament selection

In this method of selecting chromosomes, usually s strings are selected at random from a population and the best is chosen. This procedure is continued till the population size number of selections are made. This scheme can be performed with or without replacement. When performed with replacement, the best string gets more number of copies made.

(iii) Ranking Selection

Baker [10] introduced the notion of ranking selection to genetic algorithm practice. Selection by ranking method sorts the population from best to worst, assigns the number of copies that each individual should receive according to a non decreasing assignment function, and then perform proportionate selection according to that assignment. Kindly see Grefenstette and Barker[11] and Goldberg and Deb[12] for qualitative and quantitative analysis of ranking selection

2.3.5 Recombination Operators

A number of recombination operators that combine features of inversion and crossover have been investigated [7].

(i) The Crossover operator

In almost all known crossover operators two chromosomes are picked up at random and some portion of the string are exchanged them to form two new strings (children, offspring) by swapping corresponding segments of the parents The point at which the chromosomes are cut are called crossover site The fitness of the off-spring will depend upon whether the appropriate crossover site was chosen for cross over. Since the location of an appropriate site is not known in advance, a random site is chosen. Some popular crossover operators are the following -

(1)Single point crossover:- A single point crossover chooses a single crossover site randomly. The portions of the strings on one side (excluding the one at the crossover site) of the site are exchanged between parent strings.

1 1 1 | 1 1 = 1 1 1 0 0

0 0 0 | 0 0 = 0 0 0 1 1

(2)Two point crossover:- This type of cross over is affected by choosing crossover sites at random and exchanging the contents between the parent strings.

1 1 1 1 | 1 | 1 1 = 1 1 1 0 0 1 1

0 0 0 0 | 0 | 0 0 = 0 0 0 1 1 0 0

(3)Uniform crossover :-Here each bit position is exchanged between the parent strings with a probability 0 5 :

1 1 1 1 1 1 1 = 1 1 0 1 1 1 0

0 0 0 0 0 0 0 = 0 0 1 0 0 0 1

(4)PMX (Partially matched crossover):- Under this, two strings (permutation and their associated alleles) are aligned, and two crossing sites are picked uniformly at random along the strings These two points define a matching section that is used to effect a cross over through position by position exchange operations [8].

Consider two strings :

A = 9 8 4 | 5 6 7 | 1 3 2 10

B = 8 7 1 | 2 3 10 | 9 5 4 6

PMX proceeds by position wise exchange. First mapping string B to A, the 2 and 5, the 3 and the 6, and the 10 and 7 exchange places. Similarly mapping string A to B , The 5 and the 2, 6 and the 3, and the 7 and the 10 exchange places. Following PMX offspring left are A1 and B1,

A1 = 9 8 4 | 2 3 10 | 1 6 5 7

B1 = 8 10 1 | 5 6 7 | 9 2 4 3

Where each string contains ordering information partially determined by each of its parents

(5) OX(Order crossover).-[8] The order operator starts off in a manner similar to PMX Starting with the example strings a and to b to illustrate PMX. We select a matching section (for comparison, we choose the matching section of the PMX example).[8]

A = 9 8 4 | 5 6 7 | 1 3 2 10

B = 8 7 1 | 2 3 10 | 9 5 4 6

Like PMX., each string maps to constituents of the matching section of its mate. Instead of using point by point exchanges to effect the mapping a PMX does, order cross over sites uses a sliding motion to fill the holes left by transferring the mapped position. For example when B maps to string A, the jobs 5,6 and 7 will leave holes (marked by H) in the string :

B = 2 3 10 | H H H | 9 4 8 1

The holes are filled with the matching section job names taken from the mate Performing this operator and completing the complementary cross we obtain the offspring A1 and B1 as follows:

A1 = 5 6 7 | 2 3 10 | 1 9 8 4

B1 = 2 3 10 | 5 6 7 | 9 4 8 1

(6) CX (cycle crossover):-The cycle cross over scheme is very different from the PMX and OX schemes. The CX operator performs recombination in the way that each gene of the offspring comes from the corresponding position of either one of the parents To see how this works, lets start with example from C and D below: [8]

C = 2 4 5 3 8 9 6 1 7

D = 3 9 8 6 5 4 2 7 1

Cut points are not required for CX operators. Start from the left and choose a gene from the first parent :

C1 = 2 ? ? ? ? ? ? ? ?

Since every gene comes from the same location of the parents, the choice of 2 from C means we must choose 3 from the fourth position of C because of the 3 in the first position of D.

C2 = 2 ? ? 3 ? ? ? ? ?

This selection in turn requires that we select 6 from C

C3 = 2 ? ? 3 ? ? 6 ? ?

If the process continues, the choice of 6 mean we should choose 2 from C however, this is impossible because 2 has already been selected as the first gene The positions of these genes are said to form a circle that eventually returns to the first gene. To complete the operation, the remaining spaces are filled from D. the other child can be obtained by performing the complementary cross The final two offspring will be :

C1 = 2 9 8 3 5 4 6 7 1

D1 = 3 4 5 6 8 9 2 1 7

7. Greedy crossover:- This operator first picks up a random job as a starting point for the child's tour. Compares two edges leaving the starting job in the parents and choose the shorter edge. Continue to extend the partial tour by choosing the shorter of the two edges in the parents which extend the tour. If the shorter parental edge would introduce a cycle into the partial tour then extend the tour by a random edge. Continue until a complete tour is generated This crossover can be effectively used to problems like TSP. [8]

(ii) The Mutation operator

During the crossover operation some potentially useful sequences might be lost. Consequently mutation is introduced to overcome this problem. It is an

occasional random alteration process with a small probability. When mutation is applied to a sequence it chooses one bit and changes it. Here a swap operator has been used, Which randomly selects two genes and exchanges their positions in sequence. For example :

C = 2 3 5 6 7 1 4

generate two random numbers between 1 and 7 say 4 and 6, hence,

C1 = 2 3 5 1 7 6 4

2.3.6 Parameter values

Generally the determination of appropriate GA parameter setting is a trial and error process However suitable parameters can enhance the performance of GA efficiently. The following GA parameters need to be specified :

1. Number of generations,
2. Population size,
3. Probability of crossover,
4. Probability of mutation.

A higher number of first two parameters should usually produce better quality solution. The crossover and mutation probabilities are observed to perform well near the Pc = 0.95 and Pm = .001 respectively. We have used design of experiment method to get the appropriate settings of these parameters see (Chapter 3).

2.3.7 Replacement strategy: [8]

With roulette wheel selection procedure it is possible that the member of the population may fail to produce offspring in the next generation The *elitist* strategy fixes this potential source of loss by copying the best member of the population in the next generation. With this strategy, worst member of the generated population are replaced with the best member of the population.

2.3.8 Convergence policy: [8]

During the evaluation, the cost of the current best design converges more or less asymptotically to the true minimum Creating new generations and evaluating the fitness of the members should ideally be continued till the global optimum is obtained.

As the global optimum is usually unknown, another criterion for terminating the program is used.

Evaluation can be terminated, for instance, after fixed time, after certain number of generations or if there is no improvement in the last some fixed number of generations In process or product optimization problems the GA is terminated when the average fitness value of the population has reached the maximum fitness value of the population

2.4 SGA Simple Genetic Algorithm:- We are using simple Genetic algorithm coded into FORTRAN by Dr. Kalyanmoy Deb(ME IITK) as a search algorithm The code is originally taken from the book of Genetic algorithm by David E Goldberg [7] Here we will discuss some of the subroutines of the code.

Main:- This is the main part of the program. Here an initial random population of the specified size is created first by calling the *initialize* subroutine. The population is evaluated. Then one generation of GA operators· reproduction, crossover and mutation are performed on the population. Statistics for the new generation created by the operators are calculated. The results are printed by the *Report* subroutine and the new population is copied to the old generation. In the end the termination criteria is tested.

(1) Funct:- This part of the program contains the objective function for optimization. Here the subroutine is coded through which the objective function is obtained. Proper transformation is done according to the minimization or the maximization problem.

(2) Initdata :- This subroutine asks for the data to initialize GA parameters. Here the following data are entered:

1. Population size(ipopsize).
2. Length of the string (lchrom)
3. Max. no of generation (maxgen).
4. Crossover probability (pcross).
5. Mutation probability (pmute).

6. No. of parameters (nparam).

(3) Initialize:- This subroutine initializes the random number generator, creates the initial population by calling *Initpop* subroutine, calculates the statistics of the initial population and reports the initial population statistics by calling the *Initreport* subroutine

(4) Initreport:- This subroutine prints the report of the initial population. Here the maximum, minimum and average fitness values along with the data entered in *initdata* subroutine.

(5) Initpop:- This subroutine generates initial random population This is done by creating a random strings of 1 and 0 of length of chromosomes size in population size number of times

(6) Stats:- This calculates the max., min , sum and the average fitness value in each generation.

(7) Generation:- This subroutine performs the one generation of reproduction, crossover and the mutation operations This subroutine is called in the Main part of the program. Here *stochastic remainder roulette wheel* selection is done, *single point crossover* is performed by selecting a random site between the first and second bit up to just before the last bit according to the crossover probability. Occasional mutation is done. Fitness values of the first and the second child string created is calculated.

(8) Pre select:- This is primer for *stochastic remainder roulette wheel* selection it updates array 'choices' (mate pools).

(9) Iselect 2:- This function selects individuals from mating pool for stochastic remainder roulette wheel selection.

(10) Select:- This subroutine is used in simple roulette wheel selection

(11) Crossover:- This subroutine performs single point crossover between mate1 and mate2. Here first checked if crossover is to be performed or not. Then a random site is created. Then bits are exchanged between the two mates from the cross over site till the end of the string

(12) Logical not:- This function performs the logical not operation of Boolean algebra If the ivalue is 1 then it returns 0 and vice versa

(13) Mutation:- This subroutine performs mutation on ivalue probability pmute. Here iflip function is called and the ivalue is inverted if the value of iflip(pmute) tuns out to be 1.

(14) Report:- This subroutine prints the report of a generation. This writes the # of current generation Reports the old population, fitness value, newfitness value, prints the best solution, writes the best fitness value, best generation number and all the statistic of the immediate previous and the current generation.

(15) Decode:- This function decodes the binary substring of length lstr in to decimal number and returns the decoded value.

(16) X map:- This function maps the decoded value in the specified region.

(17) Decodevars:- Calculates the parameter value from a string. This is done by first decoding the value for the first parameter and then decoding for the next and so on.

(18) Copy new to old:- This subroutine copies the new population to the old population. this is done by first copying the strings, then copying the variables and in the last the fitness.

(19) Modop:- This is one of the utility subroutine. It is used to find the i mod j.

(20) Advance_random, warmup_random, random:- These are all random number initialization subroutines.

(21) Iflip:- This function simulates the flip of a coin with a given probability and 1s or 0s are generated accordingly.

(22) Irnd, Randomize:- The randomize subroutine initiates the random number generator and the irnd subroutine creates a random number between ilow and ihigh.

(Flowchart SGA- Fig 2.1).[6]

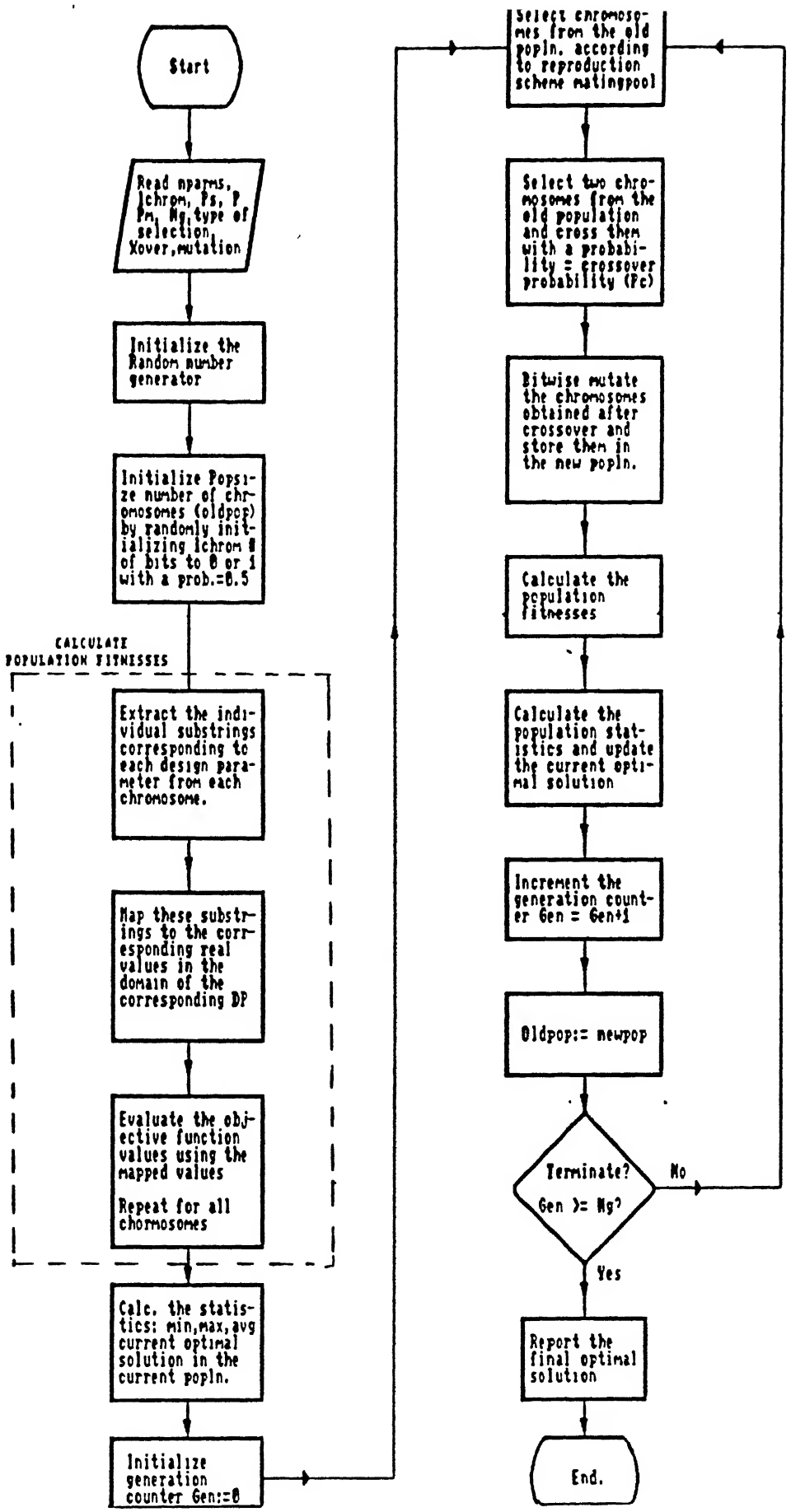


Fig-2.1 Flow chart of Simple genetic algorithm (SGA)

CHAPTER 3

THE PASSIVE FILTER DESIGN PROBLEM AND A STUDY OF ITS RESPONSES

In this chapter we will discuss the filter design problem, the results of past studies on this problem, a robust design methodology to solve this problem and the results of the this method obtained, using SGA as the search algorithm Most of the contents of this chapter repeat the work done on this problem earlier by Raman [6] However we have redone the calculations to build the computational tools required to do multi-objective robust design

3.1 Statement of the problem:- A passive filter network (Fig-3.1)[3][13][2] is to be designed to enable measurement of the displacement signal D generated by a strain-gauge transducer. The filter network provides the interface between the strain-gage transducer/demodulator and the recording instrument with a galvanometer/light-beam deflection indicator. The network conditions the signal generated by a strain-gauge transducer with demodulated output and measures the original displacement signal by filtering out the carrier (high frequency) frequency. There are two functional requirements (FRs) for the network to be designed as follows:-

(i) Minimum distortion of the output. (This is satisfied if one places the filter's pole or cutoff frequency ωc at 6.84 Hz as first functional requirement FR1.)

(ii) A full scale beam deflection D (with appropriate dc gain) of $\pm$ 3.00 inches as second functional requirement FR2 .

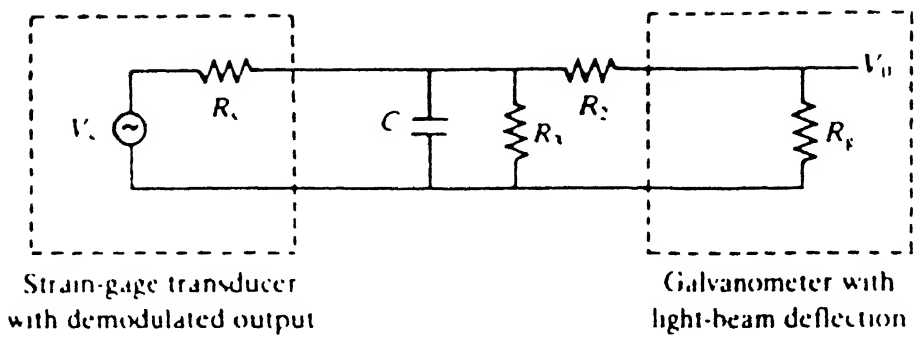


Fig 3.1 The Passive Filter Network.

The design problem here is to determine the correct set of DPs that can satisfy the two FRs The transfer function (Vo/Vs) is obtained using Kirchoff's law.

$$\frac{Vo}{Vs} = \frac{(Rg.R3)}{(R2+Rg)\,(Rs+R3)+R3\,Rs+(R2+Rg)R3.Rs.Cs} \quad \text{------ (a)}$$

Here **s** is the Laplace variable. From this transfer function, the filter cutoff frequency ωc and the galvanometer full scale deflection D may be derived as follows-

$$\omega c = \frac{(R2+Rg)(Rs+R3)+R3.Rs}{2.\Pi(R2+Rg).R3\;Rs.C} \quad \text{---------- (i)}$$

$$D = \frac{|Vo|}{Gsen} = \frac{|Vs|RgRs}{Gsen[(R2+Rg)(Rs+R3)+RsRs]} \quad \text{---------- (ii)}$$

The design parameters that the designer is free to specify are R2, R3 and C. The purchased components are Rs, Rg, Vs and Gsen Therefore we must first determine which DP among R3, R2 and C has the most influence on the FRs Next, we must find the values of those DPs that will minimize variability in the performances FR1 and FR2, when subjected to uncontrolled conditions such as component characteristic variability. If we could find two independent DPs each influencing only one FR, then we could adjust each FR to it's own target value. Here one may use orthogonal array experiments first to study the influence of each independent DP on each FR separately and then try to explore the possibility of implementation of the

Taguchi's 'two step methodology" to achieve robustness. The treatment levels for the three design parameters are:-

Table-3.1 The Treatment Levels for the Three design Parameters

		Treatment Levels	
Design Parameter	1	2	3
R3(Ω)	20	50,000	1,00,000
R2(Ω)	0.01	265	525
C(μ farad)	1,400	815	231

The reason for selecting the broad range of values in Table 3.1 is to study the nonlinear effects present if any and also sensitivity of the response to the design parameters. The broad range may help reduce the need for repeating experiments, when a high degree of noise is present

With three parameters each at three levels the total number of full factorial experiments needed to be performed are 3^3 or 27. But assuming additivity of effects, the number of experiments needed to be run becomes less. The minimum number of experiments required to be performed are determined by the degree of freedoms. The minimum number equals the total degrees of freedom Since there are three factors at three levels each the total degree of freedom are found as follows:[2]

1) 1 dof for the overall mean.

2) 3(3 -1) = 6 for the three main factors.

Therefore at least 7 experiments must be performed. The nearest standard OA (Appendix C-[1]) that accommodates three factors each at three levels is the L9 OA

Linear graph

3,4

1*----------*2

Table-3.2 The L9 OA [1]

Exp.No	Col. 1	Col. 2	Col. 3	Col.4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

We may use the L9 OA experimental setup to study up to four factors each at three levels In the present case we discard the first column of this array without affecting its utility. In Taguchi's terminology the array thus obtained is called *Inner array or control array* [1][2]as shown below -

Table-3.3 COMBINATIONS OF DP TREATMENTS IN L9 OA

Exp.No.	R3(Ω)	R2(Ω)	C(μf)
1	20	0.01	1400
2	50,000	265	815
3	1,00,000	525	231
4	20	265	231
5	50,000	525	1400
6	1,00,000	0.01	815
7	20	525	815
8	50,000	0.01	231
9	1,00,000	265	1400

To explore robustness a second OA defined as the *Outer array* would be needed to simulate the noise due to imprecision in the nominal values of the components. the effect of noise is to be simulated for each row in the inner array (L9). For noise it is anticipated that each of the purchased components Rs, Rg, Vs and Gsen vary ±0.15% around their nominal (catalogue) values. Therefore a total of seven noise sources exist in the instrumentation system using the filter network. Based on an experimental study the broadest range of DP values that are capable of fulfilling the

desired objectives viz. ; ωc = 6.84 Hz and D = 3.00 inch, are selected. The goal is to determine the variability in the output response as the combination of levels is changed over their tolerance range An exhaustive test of every combination would require 3^7 = 2,187 experiments, which is practically an impossible task. Again, the use of OAs minimizes the number of experiments while maintaining a representative sample of noise affected response Since there are in all seven noise factors with three levels each, here an L9 experiment would not suffice. An L27 OA can accommodates many as 13 control factors each at three levels. The Table 3.4 shows the noise levels for these factors

Table-3.4 Levels for Noise Factors

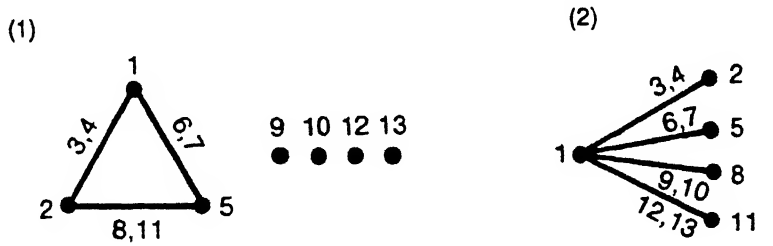
Sourse of Noise	1	Variability in Nominal value 2	3
R3(Ω)	R3 - 5%	R3	R3 + 5%
R2(Ω)	R2 - 5%	R2	R2 + 5%
C(μf)	C - 5%	C	C + 5%
Rs(Ω)	120 - 0.15% = 119.82	120	120 + 0.15% = 120.18
Rg(Ω)	98 - 0.15% = 97.853	98	98 + 0.15% = 98.147
Gsen(μ-V/in)	657.58 - 0.15% = 656.594	657.58	657.58 + 0.15% = 658.66
Vs(V)	0.015 - 0.15% = 0.014978	0.015	0.015 + 0.15% = 0.015023

The standard L27 OA that simulates the noise due to within tolerance variation as shown in the Table 3.5. The L27 OAcan accommodate upto 13 factors each at three levels. We use here the last seven columns of this array for the seven factors of Table 3.4.

Table-3.5 STANDARD L27(3^{13}) ORTHOGONAL ARRAY

Ex	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	2	2	2	2	2	2	2	2	2
3	1	1	1	1	3	3	3	3	3	3	3	3	3
4	1	2	2	2	1	1	1	2	2	2	3	3	3
5	1	2	2	2	2	2	2	3	3	3	1	1	1
6	1	2	2	2	3	3	3	1	1	1	2	2	2
7	1	3	3	3	1	1	1	3	3	3	2	2	2
8	1	3	3	3	2	2	2	1	1	1	3	3	3
9	1	3	3	3	3	3	3	2	2	2	1	1	1
10	2	1	2	3	1	2	3	1	2	3	1	2	3
11	2	1	2	3	2	3	1	2	3	1	2	3	1
12	2	1	2	3	3	1	2	3	1	2	3	1	2
13	2	2	3	1	1	2	3	2	3	1	3	1	2
14	2	2	3	1	2	3	1	3	1	2	1	2	3
15	2	2	3	1	3	1	2	1	2	3	2	3	1
16	2	3	1	2	1	2	3	3	1	2	2	3	1
17	2	3	1	2	2	3	1	1	2	3	3	1	2
18	2	3	1	2	3	1	2	2	3	1	1	2	3
19	3	1	3	2	1	3	2	1	3	2	1	3	2
20	3	1	3	2	2	1	3	2	1	3	2	1	3
21	3	1	3	2	3	2	1	3	2	1	3	2	1
22	3	2	1	3	1	3	2	2	1	3	3	2	1
23	3	2	1	3	2	1	3	3	2	1	1	3	2
24	3	2	1	3	3	2	1	1	3	2	2	1	3
25	3	3	2	1	1	3	2	3	2	1	2	1	3
26	3	3	2	1	2	1	3	1	3	2	3	2	1
27	3	3	2	1	3	2	1	2	1	3	1	3	2

Linear Graph for L27 OA:-



Corresponding to each row in the inner array, 27 orthogonally designed "outer array" experiments are conducted according to Taguchi's parameter design experimental plan (Fig-1.2 Chapter1) and the values of wc and D is calculated then for each of the 27 outer array run The mean and variance of wc and D are then calculated The Table 3.6 below shows the simulation results for one particular row (Row No.1) of inner array (Table- 3.3) with the mean and the variance of the two response characteristics as calculated The mean and variance for the other eight rows of the inner array are also shown in (Table-3.7) Then, in order to study the effect of each of the three independent parameters on the two responses the averages of the observed means and variances for one particular setting of one parameter are calculated The graphs (3.2.1-3.2.12) were observed for one particular setting of one factor while varying the other over it's entire range and then, changing the settings of the first and repeating the procedure

Table 3.6 Noise factors tested for row 1 of inner array

OA Ro #	R3(ohm)	R2(ohm)	C(farad)	Rg(ohm)	Rs(ohm)	Gsen(V/in)	Vs(V)	ωc(hz)	D(inch)
1	21.0	0105	0014700	98.1470	120 1800	0006585664	.0150225	7.1595482826	16.4245738983
2	20 0	.0100	.0014000	98 0000	120 0000	.0006575800	0150000	7 7913589478	16.6397647858
3	19 0	.0095	0013300	97 8530	119 8200	0006565936	0149775	8 5196790695	16.8613433838
4	21 0	.0100	.0014000	98 0000	119 8200	0006565936	0149775	7.5221109390	16.4146442413
5	20 0	.0095	.0013300	97.8530	120 1800	0006585664	.0150225	8.2017784119	16.6391448975
6	19.0	.0105	.0014700	98 1470	120 0000	0006575800	.0150000	7.7035999298	16.8714237213
7	21 0	.0095	.0013300	97 8530	120 0000	0006575800	0150000	7 9183540344	16.4140148163
8	20 0	0105	0014700	98 1470	119 8200	.0006565936	0149775	7.4200391769	16.6403675079
9	19 0	.0100	.0014000	97.8530	120 1800	0006585664	.0150225	8 0908460617	16 8671932220
10	19 0	0105	0014000	97.8530	120 1800	0006575800	.0149775	8.0908393860	16 8418216705
11	21 0	0100	0013300	98 1470	120 0000	0006565936	.0150225	7.9146852493	16 4708843231
12	20 0	.0095	.0014700	98 0000	119 8200	0006585664	0150000	7 4217042923	16 6118812561
13	19 0	0100	0013300	98 1470	119 8200	0006585664	.0150000	8 5160102844	16 8432636261
14	21 0	0095	0014700	98 0000	120 1800	0006575800	.0149775	7 1612143517	16 3962879181
15	20.0	0105	0014000	97 8530	120 0000	0006565936	.0150225	7 7930955887	16 6859531403
16	19 0	0095	.0014700	98 0000	120 0000	0006565936	.0150225	7 7052655220	16 9186248779
17	21 0	0105	0014000	97.8530	119 8200	0006585664	.0150000	7 5238475800	16 3861885071
18	20 0	.0100	.0013300	98 1470	120 1800	.0006575800	.0149775	8 1981086731	16 6215419769
19	20 0	.0105	.0013300	98.0000	120 1800	.0006565936	.0150000	8 1999311447	16 6677284241
20	19 0	.0100	0014700	97 8530	120 0000	.0006585664	.0149775	7.7069196701	16.8137187958
21	21 0	.0095	.0014000	98 1470	119 8200	0006575800	.0150225	7.5203795433	16 4431343079
22	20.0	.0100	.0014700	97.8530	119 8200	0006575800	.0150225	7 4233579636	16 6579551697
23	19.0	0095	0014000	98 1470	120 1800	0006565936	.0150000	8 0873718262	16 8998813629
24	21 0	0105	.0013300	98.0000	120 0000	.0006585664	.0149775	7 9165077209	16.3684997559
25	20.0	0095	0014700	98 1470	120 0000	0006585664	.0149775	7.4186930656	16 5936965942
26	19 0	.0105	.0013300	98 0000	119.8200	.0006575800	.0150225	8 5178337097	16.8901271820
27	21 0	.0100	0014700	97 8530	120 1800	.0006565936	.0150000	7 1628680229	16.4417037964

Table3.7

Row # of inner array	Avg(wc)(Hz)	Var(wc)	Avg(d)(in)	Var(d)
1	7 8002204895	0.1683026403	16.641681671	0.0358845554
2	2 1694836617	0 0087430254	0 0111114597	0.0000002781
3	6 8549427986	0.0864937678	0 0036156662	0.0000000340
4	42 144016266	5.3309350014	5.0446767807	0.0229732879
5	1 1322042942	0.0023595006	0.0072240476	0.0000001355
6	3.6215648651	0.0240658615	0.0123182768	0.0000002631
7	11.720520020	0.4186387360	2 9966247082	0.0109336907
8	12.784281730	0.2998900115	0.0246232357	0 0000010501
9	1.2618107796	0.0029576933	0 0055607506	0.0000000697

The graphs (Fig 3.2 1-3.2 12) indicate the complex interaction (Fig 1.3) effects present between the three design parameters. We recall that the designer has to seek here robustness of two performance aspects while these two responses also have to be adjusted to their respective target values If there had been no concern for noise, we could have tried to identify the optimum settings for R2, R3 and C using two equations and the inner array only but the seven possible known sources of noise led to the use of L27 outer array An attempt was made to seek robustness by maximizing the S/N ratio (minimizing the variance) and choosing the Capacitor C as adjustment parameter to bring the mean (wc) on target To adjust D the choice was not clear cut though the R2 would serve slightly better as the adjustment factor for the deflection response D, than would R3. Since the basic technology of the device being designed contained factor interaction that one could not eliminate Filippone[30] adopted a trial and error approach to produce an acceptable solution for the design problem that resulted in the following final solution:-

Soln. By	R3(ohm)	R2(ohm)	C(μf)	wc(Hz)	D(in.)	Var(wc)
Filippone	1060	463.6	257.29	6.8399	0.3396	0.0806

It is interesting to note that design obtained by Filippone using the Taguchi's Two-step procedure does not meet the two target constraints. Thus we appreciate the

complex nature of optimization being attempted. Many real life design problems present similar difficulties on the way to robustness. The limitation of the use of OAs based on additive or main factor model is exposed here. The Taguchi's two step procedure [As explained in Chapter-1] -

1 Find the setting $z = z^*$ that maximize the S/N ratio, (z being the control factors)

2 Adjust the group R to R* while keeping z fixed at z^*, to put the output on target (R being the adjustment factors).

Being tested on so many real problems, investigators have recognized that parameter optimization problem can be decomposed in to Taguchi's two step procedure for *certain product/process response* only. The search for robustness in this and certain other problems in the Electronic systems such as integrated-circuits, operational-amplifiers, active and passive filters, where we often need to satisfy more than one performance objective simultaneously need to invoke *beyond Taguchi's two step procedure*.

There has been an attempt to find a robust solution to this filter design problem by Bagchi and Kumar[13] by a method different from Taguchi's methodology. The objective of the outer array experiments is to sample the domain of noise factors so that the mean and variance of the performance responses from variations in the noise factors may be evaluated [2]. Taguchi suggested the use of outer array to sample the noise domain, the other possible approach being the use of Monte-Carlo simulation, or replication. Bagchi and Kumar adopted the same approach in view of certain particular advantages existing in it. In particular, if the noise factors do not have symmetric distribution, use of the outer array approach may bias the mean and inflate the variance estimates. Therefore, when the mathematical model linking performance to the design parameters and noise are known or may be developed (e.g. using regression technique), Monte Carlo approach should be followed. Based on the above argument they found the range of values of R3 as $100 \leq R3 \leq 350$ and correspondingly (R2 and C) which would assure the robustness of ωc . They further argued that in order for S/N_D to be maximum the value of R3 should be in the vicinity of 300. The

final choice of R3 would depend on resolving the *multiple (two) objective problem* : Maximize the robustness of cutoff frequency ωc, also maximize the robustness of deflection D. One practical approach is to identify a set of Pareto-Optimal designs. They finally said that the two extreme members of Pareto designs would be:

R3 = 300, R2 = 29, C = 454.4 μF

(this design maximizes S/Nωc) and

R3 = 200, R2 = 106, C = 424 1 μF.

(this design maximizes S/N$_D$).

R3(ohm)	R2(ohm)	C μF	Avg(ωc) (Hz)	Var(ωc)	Avg(d) (in)	Var(d)
300	29	454.4	6.845	0 089	3 006	0.011
200	106	424 1	6 847	0.0925	3.008	0.10

Both of these designs are more robust than Filippone's "Two-step" solutions. Monte-Carlo simulations require substantially more computations and effect of random trials produce fluctuations from points to points therefore number of samples should be increased to reduce these fluctuations Another method is called **constrained optimization approach** is suggested by Bagchi and Kumar, which is more effective than the Taguchi's "two-step" method and is not as expensive as doing random search Monte-Carlo simulations.

3.2 The advantages of Constrained Optimization [13]

For robust design problems such as the passive filter design problem mentioned above, the constrained optimization approach would intuitively appear to be efficient. Seeking robustness among the candidate designs which assure on-target performance in the presence of the noise factors is the essence of constrained design approach. Constrained optimization (COP) approach also makes the simulation of noise possible when the noise distribution is suspected to be asymmetric. Further, using COP, one does not have to work with signal-to-noise (S/N) ratios, this offers significant advantages over Taguchi's-two-step method. Explaining the reason for this Phadke [14] said- "Frequently as the mean decreases, the standard deviation also decreases

and vice-versa In such cases, if we work in terms of standard deviation alone, the optimization cannot be done in two steps i.e. we cannot minimize the standard deviation first and then bring the mean on target. The reason is that while one attempts to minimize the standard deviation, the mean may assume off-target value. Since in constrained optimization approach the variation is minimized considering only the on-target designs, the need for S/N ratios is eliminated". The above arguments suggest that COP approach would constitute an effective alternative strategy to the traditional "two-step Taguchi method".

3.3 Constrained optimization robust design approach[13]

Let us assume that the designer knows the different DPs and the performance Variables that he is attempting to optimize. We further assume that performance desired for characteristic Yi is identified by a target value τi. The objective is to choose DP values that will reduce the sensitivity of performance to the noise [15].

Step1. For each performance characteristic Yi to be made robust while also to be set equal to some target value τi, establish a *quantitative model* relating all design parameters{DP1, DP2, DP3..} to the performance Yi, as in·

$$\mathbf{Yi = } f\mathbf{(DP1, DP2, DP3, ----)}$$

Step2. Write the quantitative model obtained in step 1 as *constraint:*

$$f\mathbf{(\ DP1, DP2, DP3--) = \tau i}$$

If n different performances (Y1, Y2, Y3...., Yn) are being simultaneously targeted (τ1, τ2,τ3, etc.), then one would establish here a set of n constraints given by equations such as:

$$f\mathbf{(DP1, DP2, DP3, ...) = \tau 1}$$

$$g\mathbf{(DP1, DP2, DP3, ...) = \tau 2}$$

$$h\mathbf{(DP1, DP2, DP3, ...) = \tau 3}\ \text{etc}$$

Step3. Solve the equations formed in Step2 to obtain certain 'dependent' DPs in terms of the truly 'independent' DPs Clearly, if there are m DPs and n constraints with m>n, one has the choice of treating m-n DPs as independent (usable as the robust-seeking

variables in Step5 below) while the others are dependent (their values are restricted to that the design reaches the desired performance targets).

Step4. Construct the true inner array only the truly independent DPs. Also set up the appropriate outer array, or the Monte carlo experimental set-up, or the physical arrangements, for repeated observations of performance under the influence of the inner array row to observe performance(s) (Y1, Y2, Y3, ..) under the influence of noise

Step5. Apply search, response surface methods(RSMs), or some other technique to find the combination of the independent DPs that minimize the empirically estimated variance (of each performance Y1, Y2, Y3, ..) under the influence of noise. If a unique set of DP values does not optimize all performance characteristics, develop a "Pareto-optimal" set of candidate designs (Fig 1.5 1 Chapter 1)

Figure 3 1.2 below (as shown in Ref. [13][15]) pictorially conveys the essence of the constrained method for seeking robustness of several response characteristics simultaneously The quantitative models in Step 1 may be either mechanistic(based on physical laws relating the response Yi to DPi, as in the case of filter network), or it may be a regression model empirically developed that can lead to at least a second-order regression model including significant two factor interactions

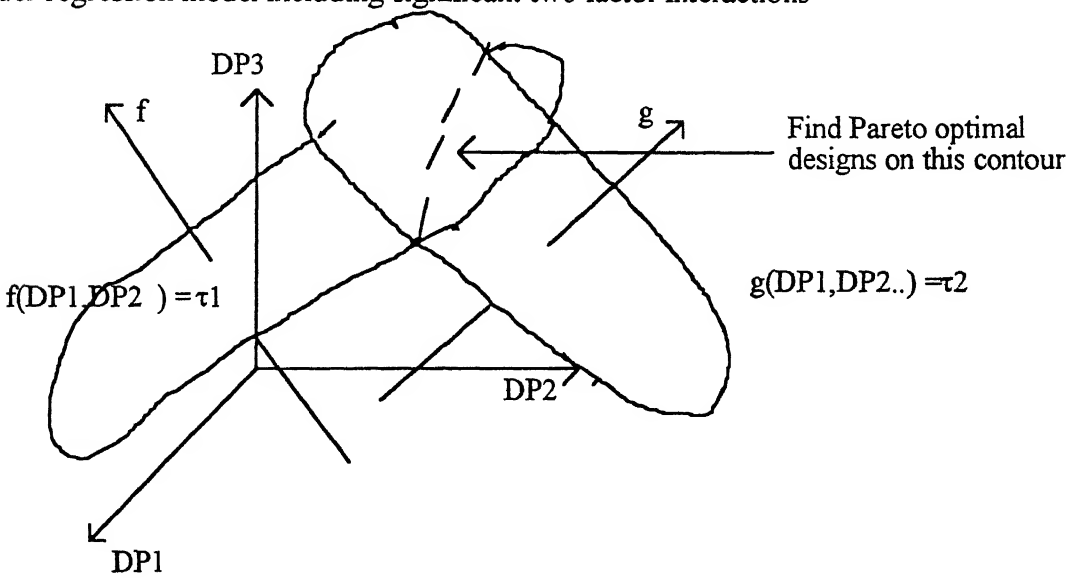


Fig 3.1.2 Pareto-optimal robust design are to be sought on the contour on which performances f and g both are on target.

3.4 Application of Constrained approach to Filter Design Problem

Now we will follow the step wise procedure as suggested in reference [13][15] to find a solution to the filter design problem using COP. The primary objective of Step 1 in the section above is to establish mathematical relationship(s) relating the performance characteristic(s) of interest to the DPs. When mechanistic models are not available from the product's or process's 'functional design', one may establish the relationship empirically For the passive filter the application of Kirchoff's laws allows us to derive these relationships (Equations (1) and (2) below) Therefore, for the passive filter problem we will directly go to Step 2.

$$\omega c = \frac{(R2 + Rg)(Rs + R3) + R3.Rs}{2.\Pi(R2 + Rg).R3\ Rs.C} \text{ ---------- (i)}$$

$$D = \frac{|Vo|}{Gsen} = \frac{|Vs|RgRs}{Gsen[(R2 + Rg)(Rs + R3) + RsRs]} \text{ ---------- (ii)}$$

These two equations are used to constrain the total design space consisting of all possible values of R3, R2 and C to the feasible set of solutions for which performance (ωc = 6.84 Hz and D = 3 in) will be *fait accompli.* Given ωc = 6.84 Hz and D = 3.00 in and a value of R3, the values of two remaining independent parameters can be obtained solving (i) and (ii) as simultaneous equations:

$$R2 = \frac{(|Vs|RgRs - DGsenR3Rs)}{DGsen(R3 + Rs)} - Rg \text{ ---------- (iii)}$$

and

$$C = \frac{|Vs|Rg(R3 + Rs)}{2\Pi\omega cR3(|Vs|RgRs - DGsenR3Rs)} \text{ ----------(iv)}$$

The combination of R3, R2 and C thus obtained is a *feasible* design. The next step is search for a feasible design which maximizes robustness Equations (iii) and (iv) together define the contour on which one needs to conduct this search. Since the filter design problem truly has *one* independent DP; once we select the value of R3, the meaningful values of R2 and C become known. The Fig-3.3(a and b) displays the Variation of two variances with the value of R3 and the Table 3.8 displays the corresponding values of R3, R2 and C. It is evident from the Table 3.8 that as soon as the value of R3 goes beyond 355 ohm the value of R2 becomes negative, this is

practically impossible to get a resistor of negative value. Therefore the range of R3 is confined in between 0 and 355 ohms.

Table 3.8

R3(ohm)	R2(ohm)	C(farad)	Avg(wc)(hz)	Var(wc)	Avg(d)(in)	Var(d)
10	580.606	0.002555	6.849897	0.153985	3.003469	0.011307
20	523.5627	0.001395	6.849122	0.142953	3.003364	0.010973
30	474.1251	0.00101	6.848455	0.134181	3.003301	0.010766
40	430.8673	0.00082	6.847878	0.127113	3.003262	0.01064
50	392.6987	0 000707	6.847371	0.121349	3.00324	0.010565
60	358.7709	0.000633	6.846921	0.116601	3.003228	0.010522
70	328.4146	0.000581	6.846521	0.112656	3.003223	0.010498
80	301.0939	0.000544	6.84616	0.109351	3.00322	0.010485
90	276.3751	0.000515	6.845831	0.106564	3.003221	0.010478
100	253.9035	0.000493	6.84553	0.104197	3.003221	0.010475
110	233.386	0.000476	6.845252	0.102177	3.003222	0.010474
120	214.5782	0.000462	6.844991	0.100441	3.003224	0.010473
130	197.2751	0.000452	6.844748	0.098942	3.003226	0.010472
140	181.303	0.000443	6.844518	0.097641	3.003227	0.010472
150	166.514	0.000437	6.844297	0.096505	3.00323	0.010473
160	152.7813	0.000432	6.844085	0.095507	3.003234	0.010475
170	139.9958	0.000429	6.843884	0.094624	3.003237	0.01048
180	128.0626	0.000426	6.843686	0.093838	3.003241	0.010488
190	116.8993	0.000425	6.843495	0.093134	3.003247	0.010499
200	106.4337	0.000424	6.843308	0.092497	3.003254	0.010515
210	96.60234	0.000424	6.843124	0.091917	3.003262	0.010536
220	87.34933	0.000425	6.842945	0.091385	3.003273	0.010562
230	78.62506	0.000427	6.842768	0.090891	3.003285	0.010596
240	70.38548	0.000429	6.842594	0.090431	3.0033	0.010636
250	62.59126	0.000432	6.842422	0.09	3.003317	0.010684
260	55.20731	0.000435	6.842255	0.089594	3.003337	0.010741
270	48.20198	0.000439	6.842092	0.08921	3.003361	0.010807
280	41.54694	0.000444	6.841934	0.08885	3.003386	0.010882
290	35.21652	0.000449	6.841781	0.088513	3.003415	0.010967
300	29.18755	0.000454	6.841636	0.088202	3.003448	0.011062
310	23.439	0.000461	6.841499	0.087923	3.003484	0.011168
320	17.95175	0.000467	6.841375	0.087682	3.003523	0.011286
330	12.70838	0.000475	6.841267	0.087489	3.003566	0.011415
340	7.692986	0.000482	6.841173	0.087358	3.003613	0.011556
350	2.891014	0.000491	6.841103	0.087303	3.003665	0.01171
360	-1.71089	0.0005	6.841057	0.087347	3.00372	0.011876
370	-6.12495	0.00051	6.841043	0.087515	3.003778	0.012055
380	-10.3625	0.000521	6.841067	0.08784	3.003842	0.012247
390	-14.4338	0.000532	6.841138	0.08836	3.003909	0.012453
400	-18.3485	0.000544	6.841264	0.089127	3.003982	0.012673
410	-22.1155	0.000557	6.841455	0.090199	3.004058	0.012906

420	-25.743	0.000571	6.84173	0.091652	3.004138	0.013154
430	-29.2386	0.000586	6.842102	0.093576	3.004225	0.013417
440	-32.6093	0.000603	6.84259	0.096085	3.004315	0.013693
450	-35.8618	0.00062	6.843219	0.099317	3.00441	0.013985
460	-39.0021	0.000639	6.844024	0.103446	3.004509	0.014292
470	-42.036	0.000659	6.845033	0.108685	3.004614	0.014614
480	-44.9687	0.000681	6.846294	0.115304	3.004722	0.014951
490	-47.8053	0.000705	6.847863	0.123641	3.004837	0.015304
500	-50.5504	0.000731	6.849808	0.13412	3.004956	0.015673

Since here we have only one independent DP, so the conduct of *inner array* greatly simplifies in this case. The step5 is accomplished by performing the *outer-array* using the noise conditions consistent with the Table-(3.4) Here we can even use Monte-Carlo simulations but since the noise distribution is symmetrical, the outer-array produces results comparable with Monte-Carlo simulations. The choice of R3 is guided by the constraint that the values of R2 and C obtained with eqns (3) and (4) should be ≥ 0. The last step towards robustness is to apply search methods to find the combination of independent DPs that minimizes the empirically estimated variance under the influence of noise. The most traditional optimization methods used in the design optimization can be divided in two broad classes: direct search methods and the gradient (descent) search methods The selection of a particular method depends upon a number of considerations mentioned by Rao [16] as follows:

(i) The type of problem to be solved (ii) availability of a ready made computer program (iii) the necessity of the derivatives of the objective function and the constraints (iv) available knowledge of the efficiency of the method (v) the accuracy of the solution desired (vi) ability of the method in finding the true (global) optimum (vii) the generality of the method in solving other problems (viii) the ease of use of the program.

Direct search methods [6] require only the objective function values, whereas the gradient search methods require the gradient information One common property of these optimization algorithm s is that they all work by point to point basis. Initializing either by the program or randomly or at times guided by the physical or

geometrical aspects of the problem. Essentially different optimization algorithms vary according to the transition rule they employ to find a new point. In general, direct search methods are computationally expensive and in most cases seem to work effectively on simple unimodal functions. Gradient based search methods, on the other hand require the knowledge of the gradients of the objective functions and constraints. These methods too do not guarantee the global solution. The other problem is that in robust design problems the objective function is not known explicitly (where ever it is calculated using outer-array or Monte-Carlo simulation), thus an exact computation of gradient may not be possible in some cases Besides, some Random Search techniques are also used where no knowledge about the problem is available or if objective function is discontinuous and non differentiable or the search space is too large. These methods though not very efficient are used at early stages of the optimization to detect the regions where the global optimum is likely to be found.[6]

In general we need a robust search technique that can applied to wide variety of problems. In robust design, one encounters a variety of objective functions and discrete as well as continuous design variables together with multiple responses This requires a great deal of problem specific knowledge, experience and judgment. Recent literature [17-21] reports that Genetic -Algorithms can be effective in wide variety of search and optimization problems in science, engineering and commerce. Here we are using Genetic-Algorithm based search to find the best value of R3.

3.5 SELECTION OF BEST GA PARAMETERS [6]

The GAs strike a good balance between exploration of the total search space and exploitation of good solutions currently at hand Two important aspects guide the evolution process of the "genetic search", namely, population diversity and selective pressure. An increase in the selective pressure decreases the diversity and vice versa. Concentrating the search on the top individuals (exploitation), the genetic diversity is lost. On the other hand, reducing the selective pressure increases the exploration. In other words strong selective pressure "supports" the premature convergence while weak selective pressure can make the search ineffective. Thus it becomes important to

strike balance between these two factors by suitably selecting the GA control parameters namely; string length, population size, the probability of crossover, the probability of mutation, etc.

Another important consideration that influences the genetic search is the representation structure Representation structure defines the genetic operators which pass some characteristic structure from parent to offspring. The representation structure so chosen that the schemata are meaningful and relevant to the problem and the GA operators are able to juxtapose the underlying building block

After having implemented a suitable representation structure the decision regarding the string length is largely based on the accuracy desired in the solution and the number of design variables. In general, for binary-coded GAs the accuracy expected of string l is of the order[22]

$$O((xmax - xmin)/2^{l})$$

The next set of decisions to be made regarding the reproduction scheme, crossover probability, and the mutation probability. The reproduction is responsible for exploiting the current population by making the duplicates of good strings preventing the passing of bad strings in the next generation Crossover and mutation operators are responsible for exploring a set of (hopefully) good strings selected by reproduction operator, for better strings [6]

In a recent study Deb and Goldberg [23] has shown that in order for the GAs to work on simple bit-wise linear problems, the following inequation should be satisfied,

$$p_c \geq \frac{(e^{l/n})\log S}{n \log n}$$

Where n in the population size, S is the selection pressure and p_c is the cross over probability Goldberg has derived a count of unique expected schemata for the strings of given length and population of given size. Based on this count Goldberg derived a rational problem-independent, performance measure for selecting optimal population sizes. This performance measure (the number of expected schemata per

population member) is maximized using Fibonacci search. The empirical fit of the results has given an approximate relationship for the optimal population size as a function of the string length,

$$m^* = 1.65 \times 2^{0.211}$$

valid for string length upto 60. Aside from the above, however there exists no guidelines as to what combination of reproduction scheme, population size Ps, crossover probability Pc and mutation probability Pm be used for different optimization problems Since the choice of parameters have a considerable influence on the performance of genetic search, we have experimentally studied the effect of the three key GA parameters viz. Ps, Pc and Pm and tried to find the optimum set these three parameters to reach an objective function value close to the true optimum with smallest number of function evaluations

3.6 EXPERIMENTAL SETUP FOR SELECTION OF BEST GA PARAMETERS

The Simple Genetic algorithm (SGA) is used to study the effects of the three main GA factors. The three factors Ps, Pc and Pm are studied at three different levels each. A full factorial experimental setup is chosen to carry out an exhaustive study. All other GA parameters viz.; length of string (l), number of generations (maxgen), seed random number are kept fixed for each of the 27 experiments. The Minimum and Maximum bounds for the single independent parameter R3 is kept such that other two parameters, R2 and C have positive values. Here it should be noted that as soon as we increase the value of R3 greater than 355 the value of R2 becomes negative which is possible for any practical resistor [6].

SGA parameters:-

1. No of parameters (nparam) = 1
2. No of generations (maxgen) = 10
3. Length of string (lchrom) = 32
4. Min. and Max. bound for R3 = 0 355.

Table 3.9 Factor levels for SGA parametric study[6]

Levels	Ps	Pc	Pm
1	25	0 75	0.1
2	50	0.90	0.05
3	100	1.0	0.01

Factor levels for SGA parametric study expressed in terms of the string length l:

Table 3.10

Levels	Ps	Pc	Pm
1	0 78125*l	0.75	3.2/l
2	1.5625*l	0 90	1.6/l
3	3.125*l	1.00	0.31/l

Table 3.11 3x3x3 EXPERIMENTAL SETUP

Exp #	Ps	Pc	Pm
1	10	0.75	0.1
2	10	0.75	0.05
3	10	0 75	0.01
4	10	0 90	0.1
5	10	0 90	0.05
6	10	0.90	0 01
7	10	1.0	0.1
8	10	1 0	0.05
9	10	1 0	0 01
10	40	0.75	0.1
11	40	0.75	0.05
12	40	0.75	0.01
13	40	0.90	0.1
14	40	0.90	0.05
15	40	0 90	0.01
16	40	1.0	0.1
17	40	1 0	0.05
18	40	1.0	0 01
19	100	0 75	0.1
20	100	0.75	0 05
21	100	0 75	0.01
22	100	0.90	0.1
23	100	0.90	0.05
24	100	0.90	0.01
25	100	1.0	0.1
26	100	1.0	0.05
27	100	1.0	0.01

The array used in the study of parameters is called full factorial *inner array* The only noise variable here is seed random number used in initializing the program execution The termination criteria used here is fixed number of generations = 10 We will increase the number of generations as we identify the best combinations of GA parameters under study Sometimes a termination criteria of fixed number of function evaluations for all experimental runs is also used In this case the program runs upto different number of generations for different combinations of Pc and Pm for fixed no of function evaluations For example if a termination criteria of 5000 function evaluations is used then for Pc = 0 75 and Ps = 50, the program will run upto,

$$Ng = 5000/0\ 75^{*}\ 50 = 133\ 3 \sim 134 \text{ Generations}$$

Here we are using a termination criterion of fixed number of function evaluations Therefore the SGA will run upto fixed number of generation as specified in the input parameters for SGA [6]

3.7 The Experimental results and their interpretation

The selection of the best GA parameters is done using the full factorial experimental plan, with the parametric values as shown in the table 3 9 Here we are studying the parameter effects on variance of ωc because it more sensitive of the two variances, i e the magnitude of the Var(wc) is more compared to Var(d) Each experiment is run at five values of the seed random number and the values obtained are observed Table 3 12 displays the results of the runs and Table 3 13 shows the Average, Maximum, Minimum and the Range of the values for five replicates of each experiment

Table 3.12 MINIMUM VALUE OF Var(ωc) OBTAINED FOR THE FIVE REPLICATES OF THE INNER ARRAY RUNS

Repli-cate	1	2	3	4	5
Exp #	Seed=0 2	Seed=0 3	Seed=0 4	Seed=0 45	Seed=0 5
1	0 087303639	0 087303524	0 087307692	0 08730257	0 08730387
2	0 087303753	0 087399721	0 087302685	0 087306614	0 08730697
3	0 087302799	0 087419272	0 087484951	0 087305546	0 08804837
4	0 087302799	0 087303524	0 087306385	0 087305183	0 08730292
5	0 087303524	0 087399845	0 087302799	0 087305183	0 08738636
6	0 087303524	0 087391014	0 087303877	0 087305183	0 08804857
7	0 087302685	0 087302799	0 087344408	0 087306976	0 08730661
8	0 087303047	0 087304478	0 087312107	0 087304707	0 08730459
9	0 087303639	0 087355251	0 087482929	0 087303877	0 08730459
10	0 087303276	0 087302685	0 08730257	0 087303047	0 08730257
11	0 087302685	0 087302799	0 08730257	0 087302685	0 08730268
12	0 087302685	0 087302799	0 087302799	0 087305183	0 08730268
13	0 08730257	0 087302685	0 08730257	0 087302799	0 08730268
14	0 087302685	0 08730257	0 087302446	0 08730257	0 08730257
15	0 087303276	0 087302685	0 087302685	0 087305183	0 08730268
16	0 08730257	0 087302799	0 08730257	0 087302685	0 08730268
17	0 087302685	0 087302685	0 087302799	0 087302685	0 08730268
18	0 087303276	0 087302685	0 08730257	0 087305183	0 08730268
19	0 08730257	0 08730257	0 08730257	0 08730257	0 08730257
20	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
21	0 087302685	0 08730257	0 08730257	0 087302685	0 08730257
22	0 08730257	0 087302685	0 087302685	0 08730257	0 08730257
23	0 08730257	0 087302685	0 08730257	0 08730257	0 08730257
24	0 087303162	0 08730257	0 087302685	0 08730257	0 08730244
25	0 08730257	0 087302685	0 087302685	0 08730257	0 08730257
26	0 087302685	0 08730257	0 08730257	0 08730257	0 08730257
27	0 08730257	0 087302446	0 08730257	0 087302799	0 08730257

Table 3.13 AVERAGE, MAXIMUM ,MINIMUM AND RANGE OF THE MINIMUM Var(ωc) FOR FIVE REPLICATES OF EACH EXPERIMENT

Exp #	Average	Max	Min	Range
1	0 08730426	0 087307692	0 08730257	5 1217E-06
2	0 08732395	0 087399721	0 087302685	9 7036E-05
3	0 087512188	0 088048373	0 087302799	0 000745573
4	0 087304163	0 087306385	0 087302799	3 5858E-06
5	0 087339544	0 087399845	0 087302799	9 70459E-05
6	0 087470434	0 088048573	0 087303524	0 000745049
7	0 087312696	0 087344408	0 087302685	4 17229E-05
8	0 087305786	0 087312107	0 087303047	9 0599E-06
9	0 087350058	0 087482929	0 087303639	0 000179291
10	0 08730283	0 087303276	0 08730257	7 058E-07
11	0 087302685	0 087302799	0 08730257	2 289E-07
12	0 08730323	0 087305183	0 087302685	2 4986E-06
13	0 087302662	0 087302799	0 08730257	2 289E-07
14	0 087302568	0 087302685	0 087302446	2 388E-07
15	0 087303303	0 087305183	0 087302685	2 4986E-06
16	0 087302662	0 087302799	0 08730257	2 289E-07
17	0 087302708	0 087302799	0 087302685	1 144E-07
18	0 08730328	0 087305183	0 08730257	2 6131E-06
19	0 08730257	0 08730257	0 08730257	0
20	0 087302545	0 08730257	0 087302446	1 243E-07
21	0 087302616	0 087302685	0 08730257	1 145E-07
22	0 087302616	0 087302685	0 08730257	1 145E-07
23	0 087302593	0 087302685	0 08730257	1 145E-07
24	0 087302687	0 087303162	0 087302446	7 156E-07
25	0 087302616	0 087302685	0 08730257	1 145E-07
26	0 087302593	0 087302685	0 08730257	1 145E-07
27	0 087302591	0 087302799	0 087302446	3 532E-07

Observing the Table 3 13 and the factorial effect graphs (Fig-3 4a -3 4c) we find that for the population size of 10 that is for Experiment No 1-9, the value of the variance is minimum at the crossover probability of Pc =1 0 and Pm = 0 05, therefore at the population size of 10 the best parameter setting is

Pc = 1 0 and Pm = 0 05 ,

This corresponds to the Experiment # 8 (Table-3 11)

-For the population size of 40, that is for Experiment No 10-18 the value of variance is minimum at Pc = 0 9 and Pm = 0 05, therefore the best parameter setting for medium size population (Ps = 40) is

Pc = 0 9 and Pm = 0 05,

This corresponds to the Experiment # 14 (Table-3 11)

-For the population size of 100 the best solution is obtained at the experiment # 20 This corresponds to the parameter settings of Pc = 0 75 and Pm = 0 05 (Table-3 11) Here one thing is noted that the average of the best solution obtained for the population size of 10 is not as good as that for population the population size of 40 Same is between population size of 100 and 40 The overall best solution is obtained at the population size of 100 (Fig 3 5a-3 5c) Here the value of the average Variance is considerably less than that found at other two population sizes (10 and 40) The range of the maximum and the minimum is also very close to the minimum at these parameter settings (Fig 3 6) The way the solutions converge towards the best point with generations for the Experiment # 20 is shown in the Fig-3 7 for five different values of the seed random number The consistency of GA's convergence at Ps = 100, Pc = 0 75 and Pm = 0 05 is evident from Figure 3 7

Fig 3.2.1 Mean wc Vs R3R2

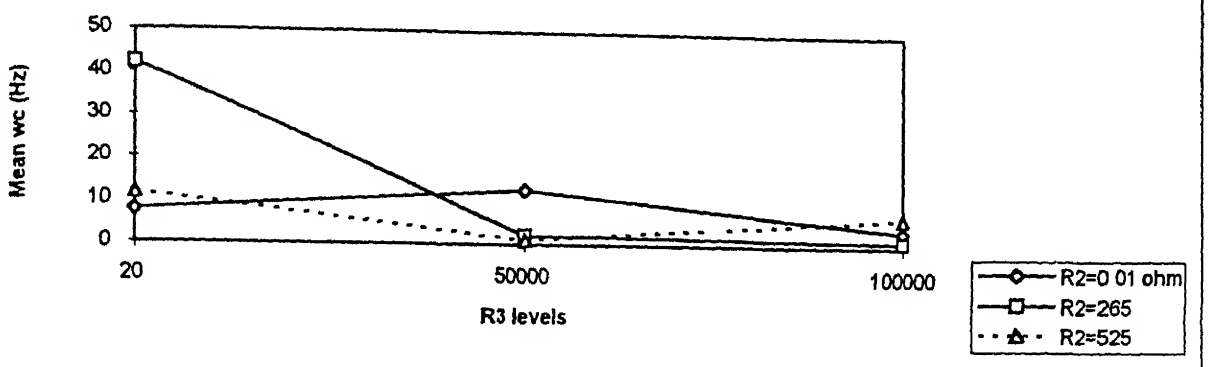


Fig 3 2 2 Mean wc Vs R3C

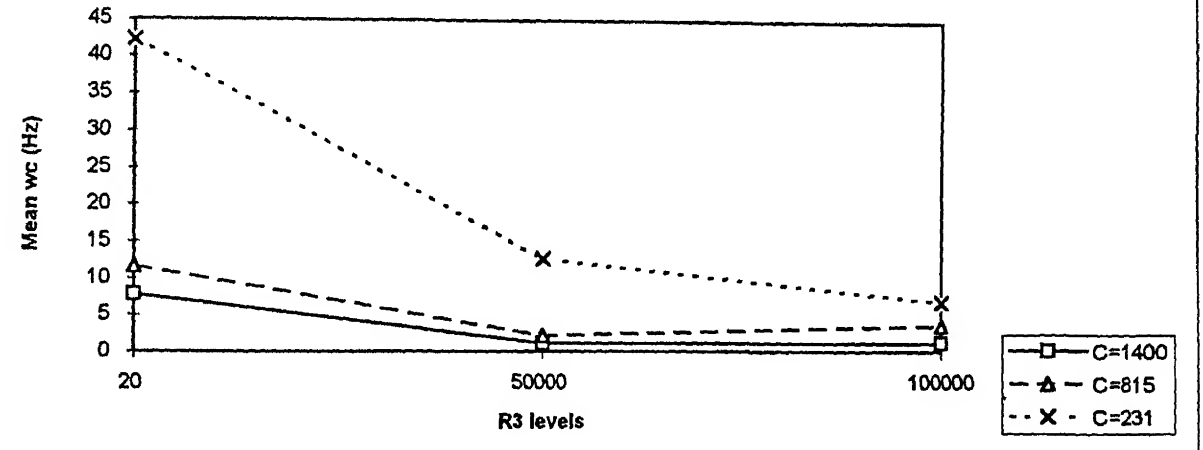


Fig 3.2 3 Mean wc Vs CR2

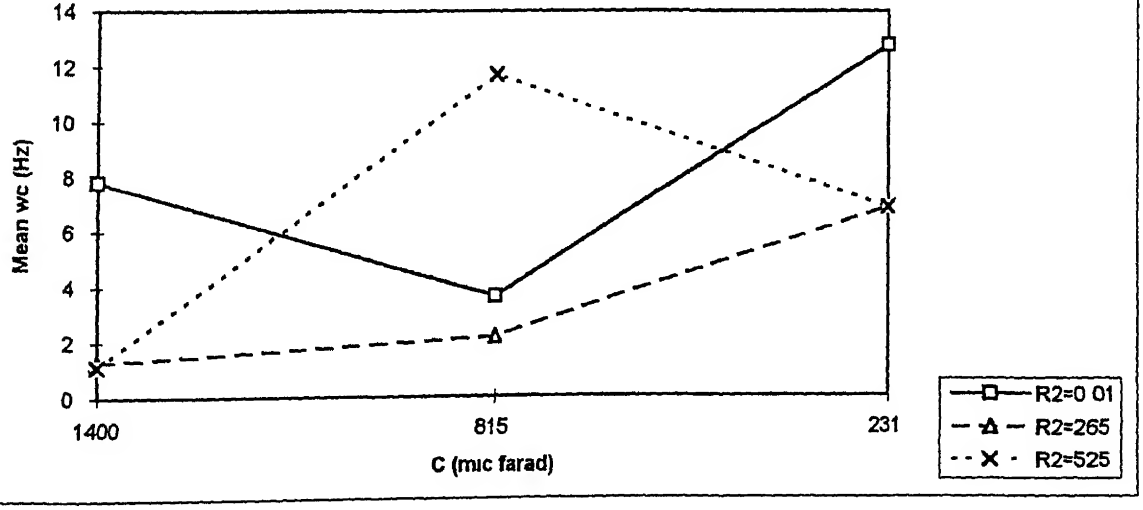


Fig 3 2.4 Variance of wc Vs R3 R2

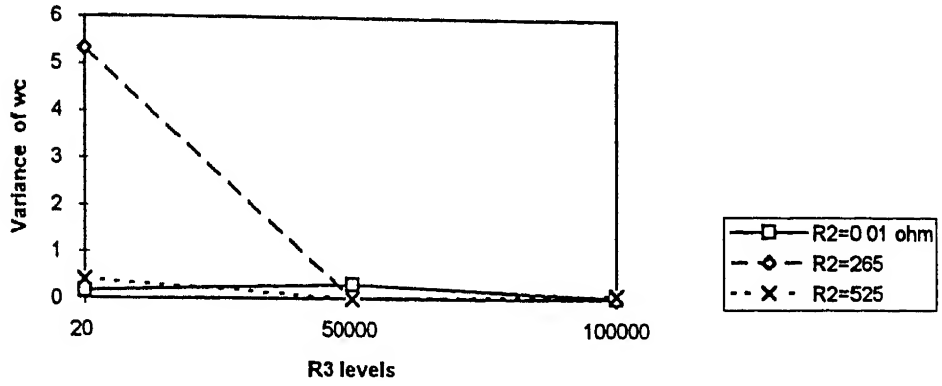


Fig 3 2.5 Variance of wc Vs R3C

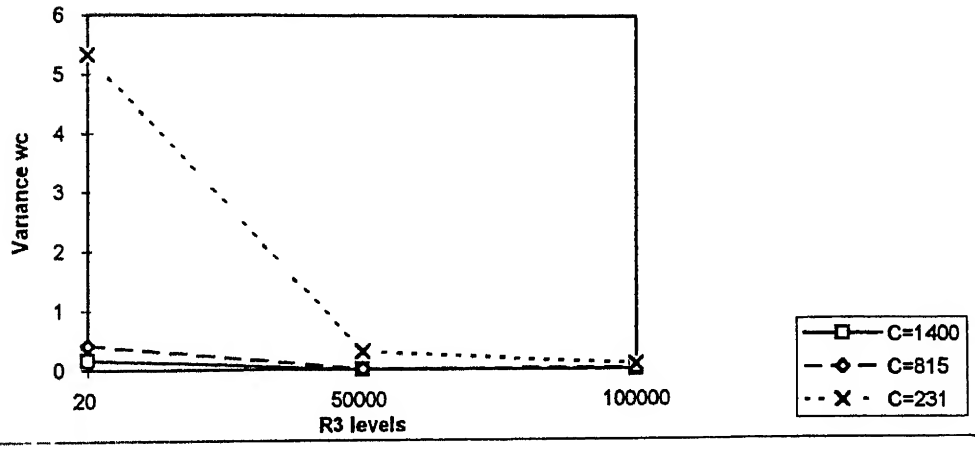


Fig 3 2.6 Variance of wc Vs CR2

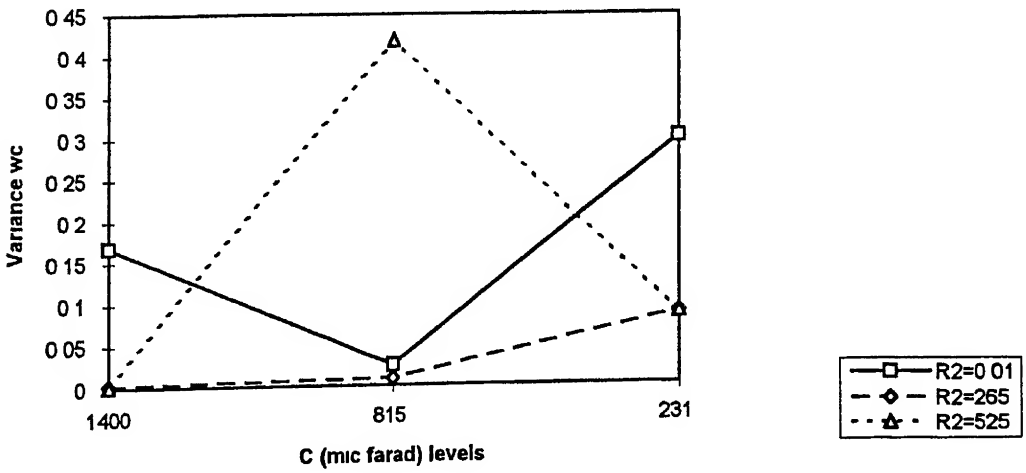


Fig 3 2.7 Variance od D Vs R3R2

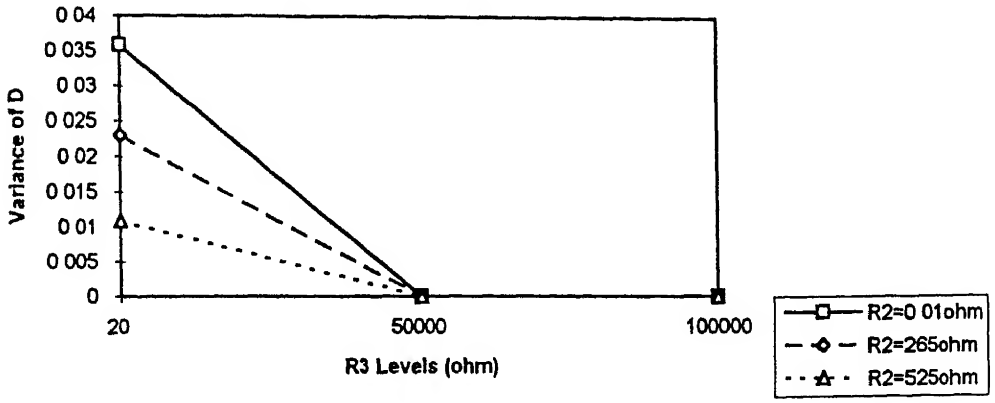


Fig 3.2.8 Variance of D Vs R3C

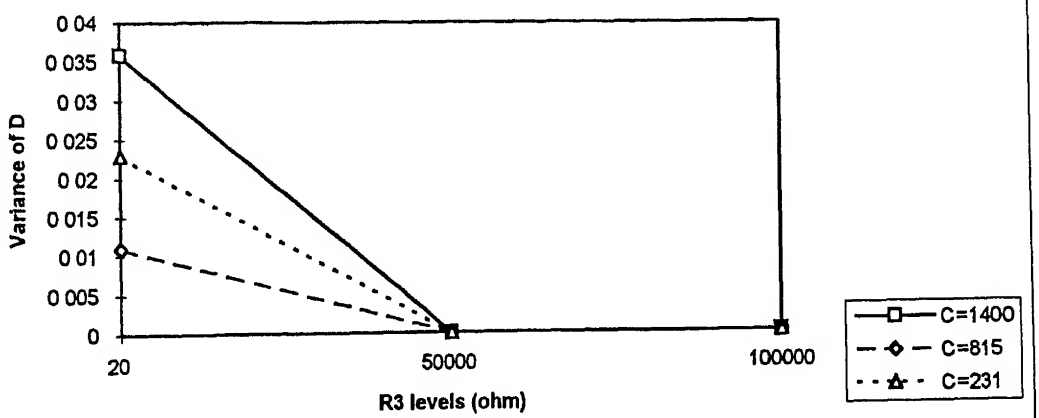


Fig 3.2 9 Variance of D Vs CR2

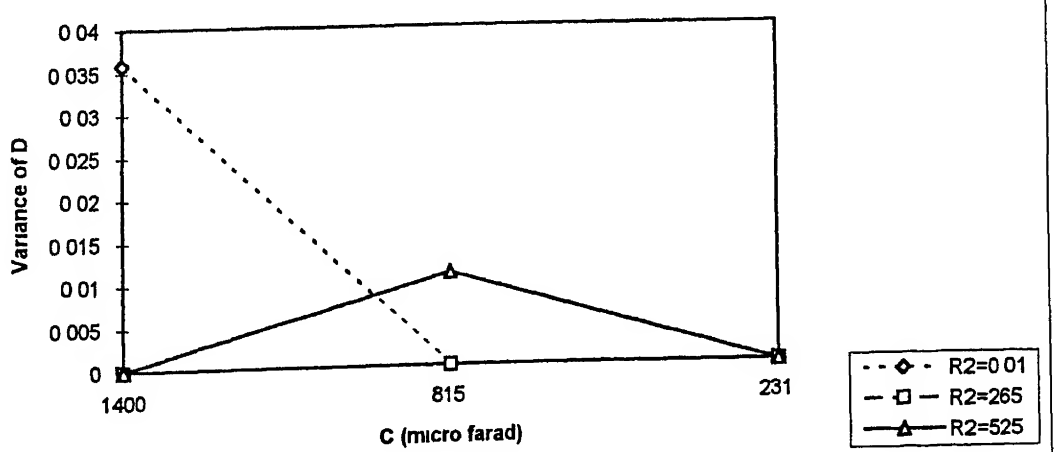


Fig 3.2.10 Mean D Vs R3R2

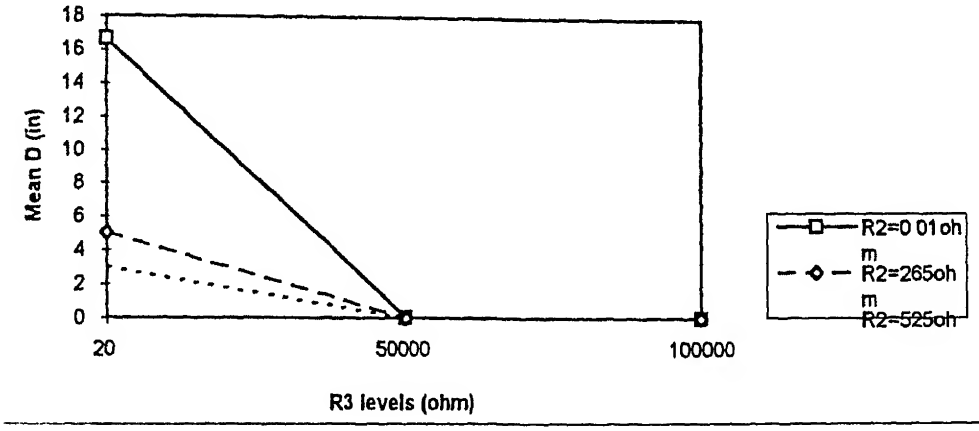


Fig 3 2 11 Mean D Vs R3C

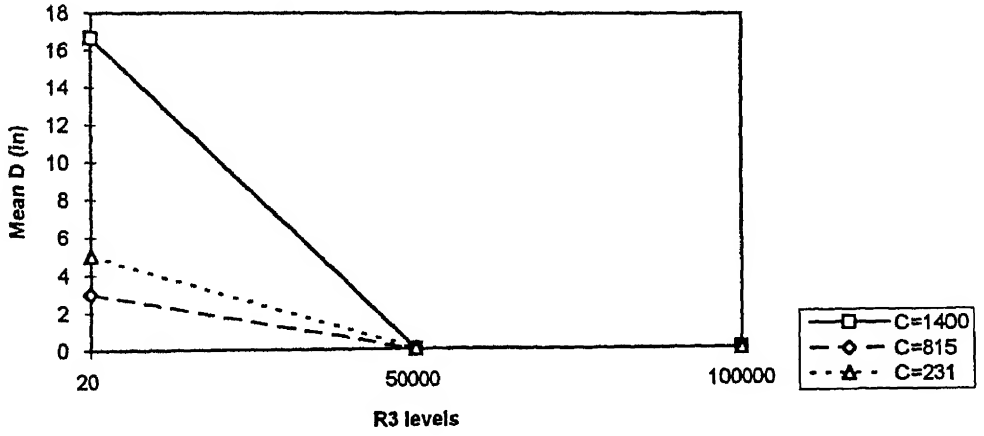


Fig 3.2.12 Mean D Vs CR2

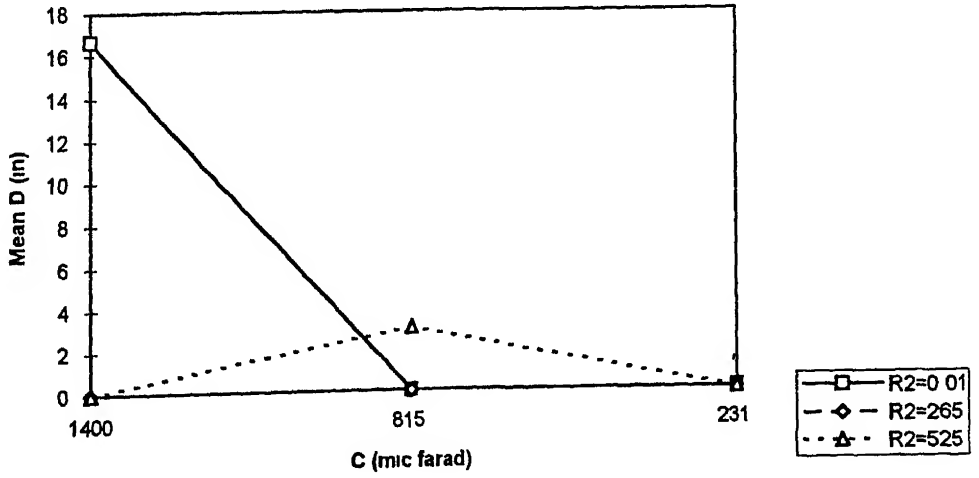


Fig-3.3 a Variation of Var(wc) with R3

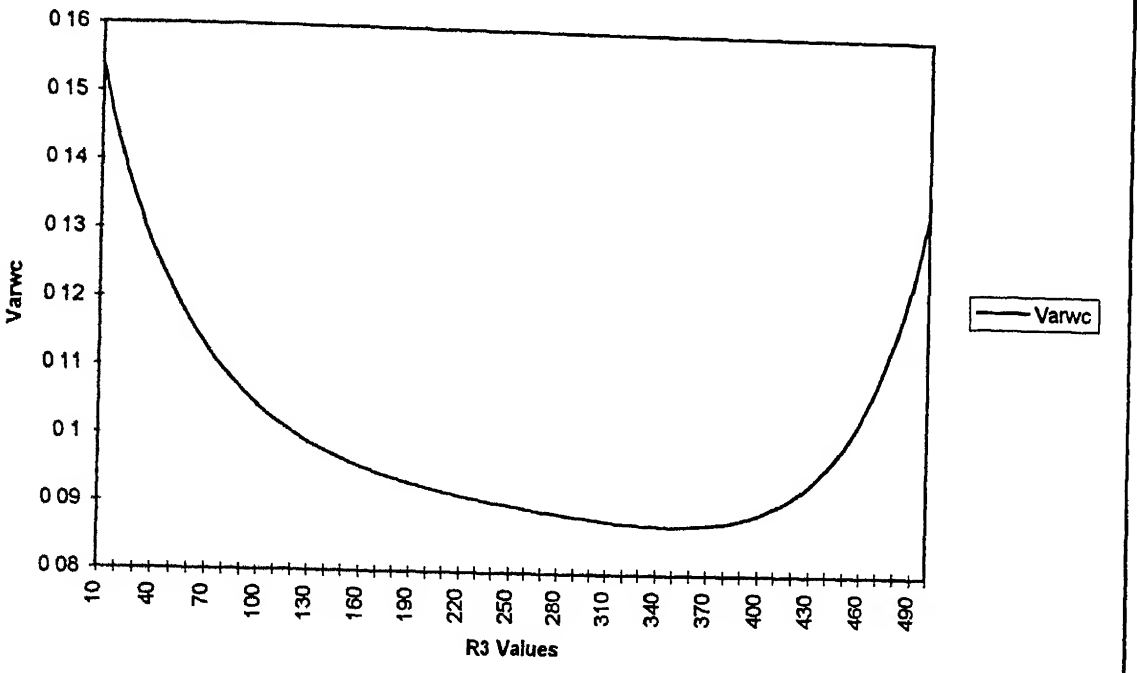
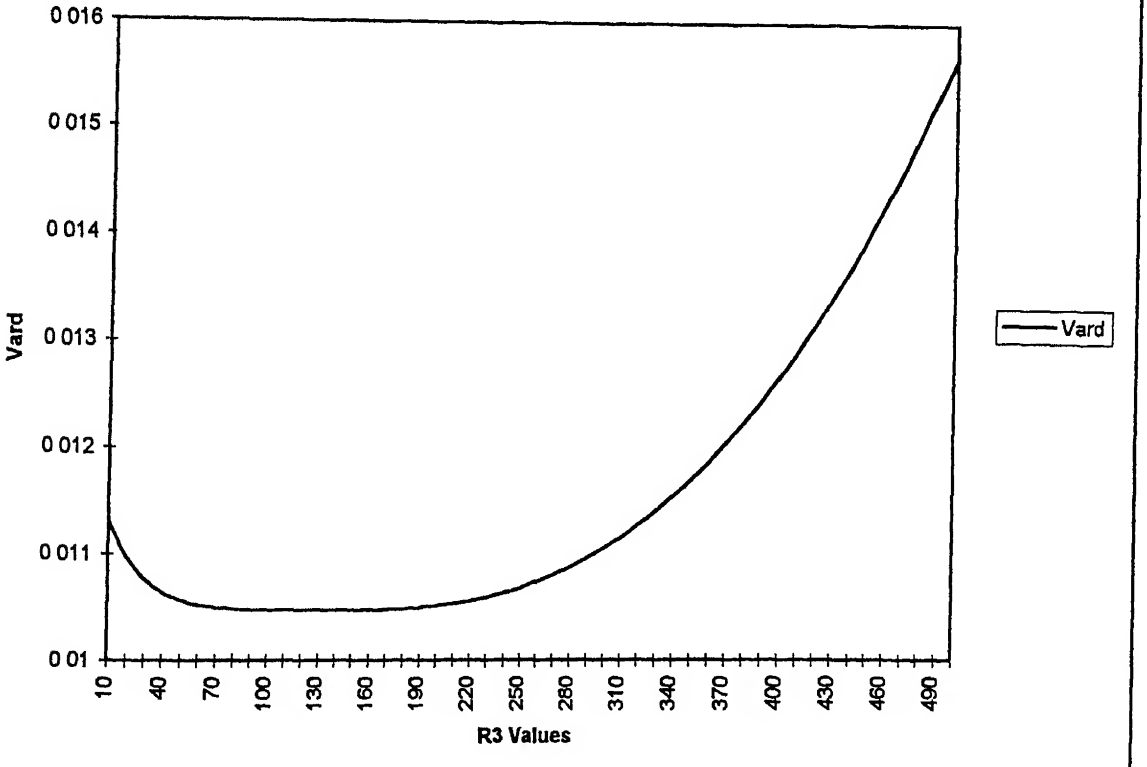


Fig3 3b Variation of Var(d) with R3 values



Ps	Pm	Pc=0 75	Pc=0 9	Pc=1 0
10	0 1	0 08730426	0 087304163	0 087312696
10	0 05	0 08732395	0 087339544	0 087305786
10	0 01	0 087512188	0 087470434	0 087350058
Ps	Pm	Pc=0 75	Pc=0 9	Pc=1 0
40	0 1	0 08730283	0 087302662	0 087302662
40	0 05	0 087302685	0 087302568	0 087302708
40	0 01	0 08730323	0 087303303	0 08730328
Ps	Pm	Pc=0 75	Pc=0 9	Pc=1 0
100	0 1	0 08730257	0 087302616	0 087302616
100	0 05	0 087302545	0 087302593	0 087302593
100	0 01	0 087302616	0 087302687	0 087302591

Fig 3.4a Objective function value (Variance wc) Vs combinations of Pc and Pm at Ps=10

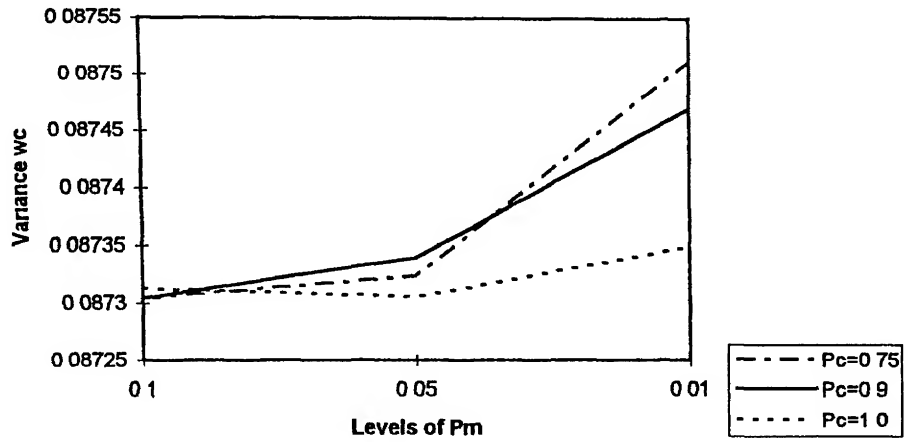


Fig.3.4b Variance wc Vs combinations of Pc and Pm at Ps=40

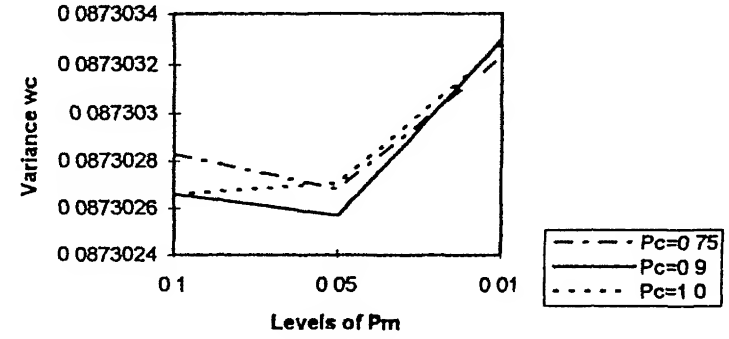


Fig.3.4c Variance wc for Vs combinations of Pc and Pm at Ps=100

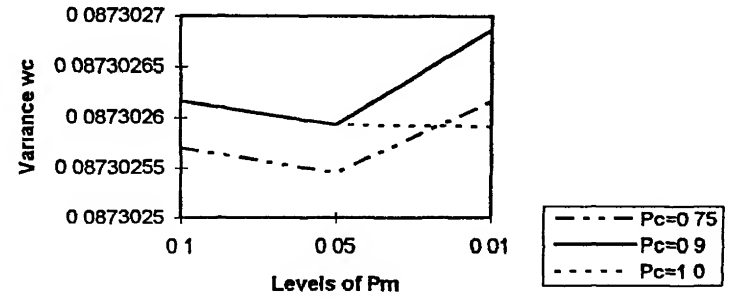


Fig 3.5c Minimum, Maximum and Average value of objective function obtained for the five replicates of the inner array runs.

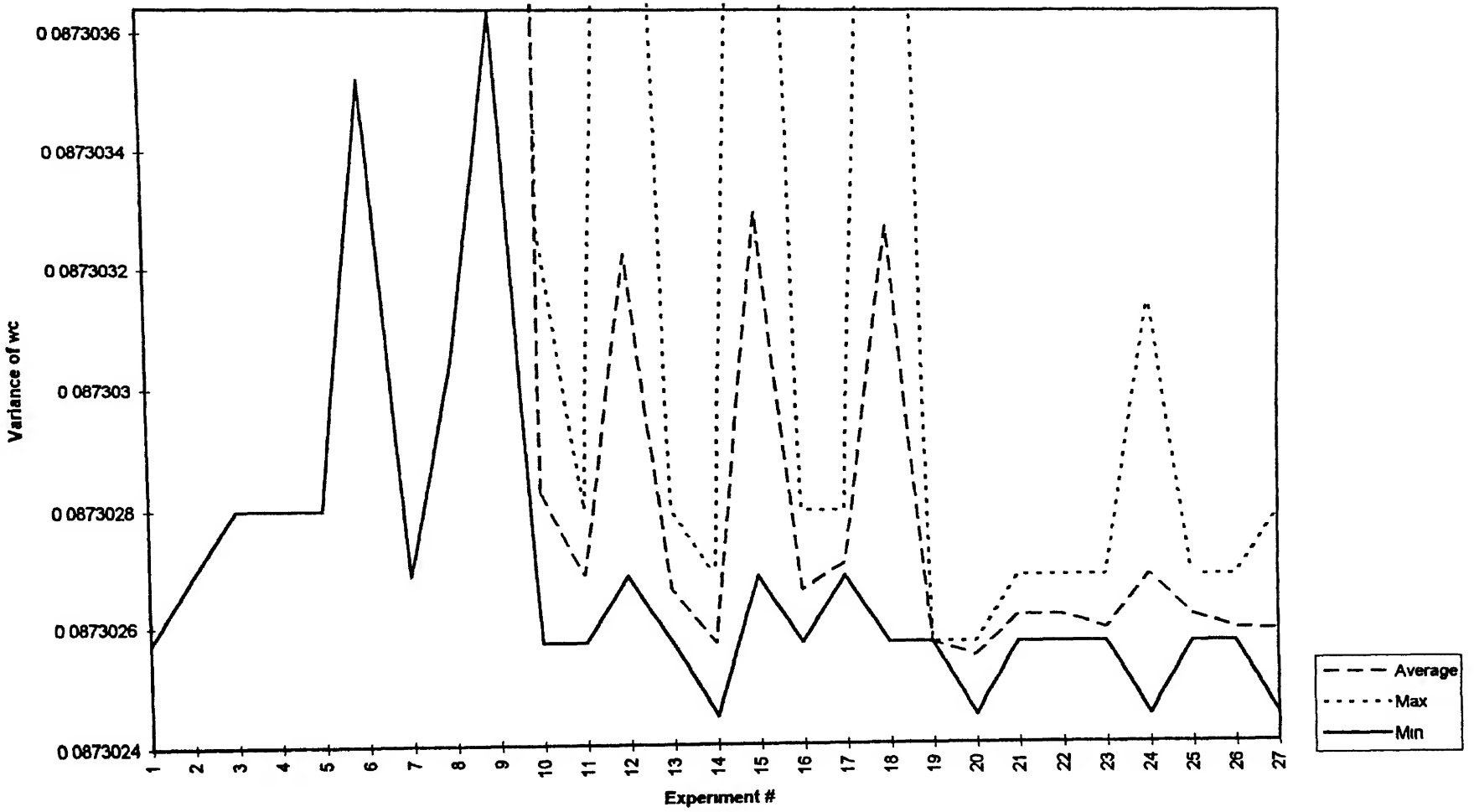


Fig 3.6a Range of Maximum and minimum value of Average variance of wc obtained in inner array runs

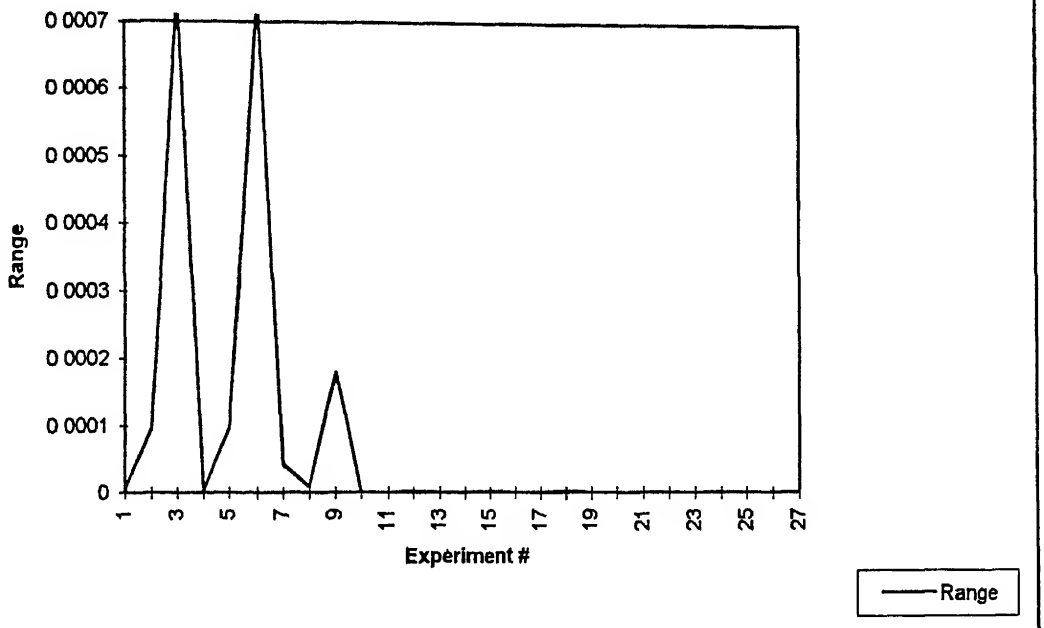


Fig 3 6bThe range of maximum and minimum value of Average variance of wc in inner array experiments

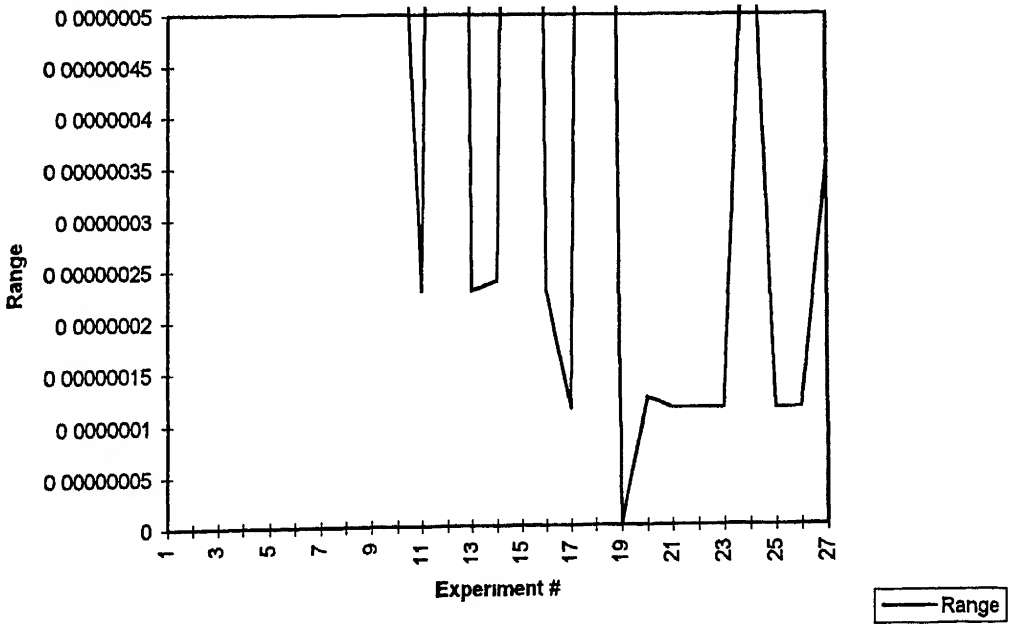
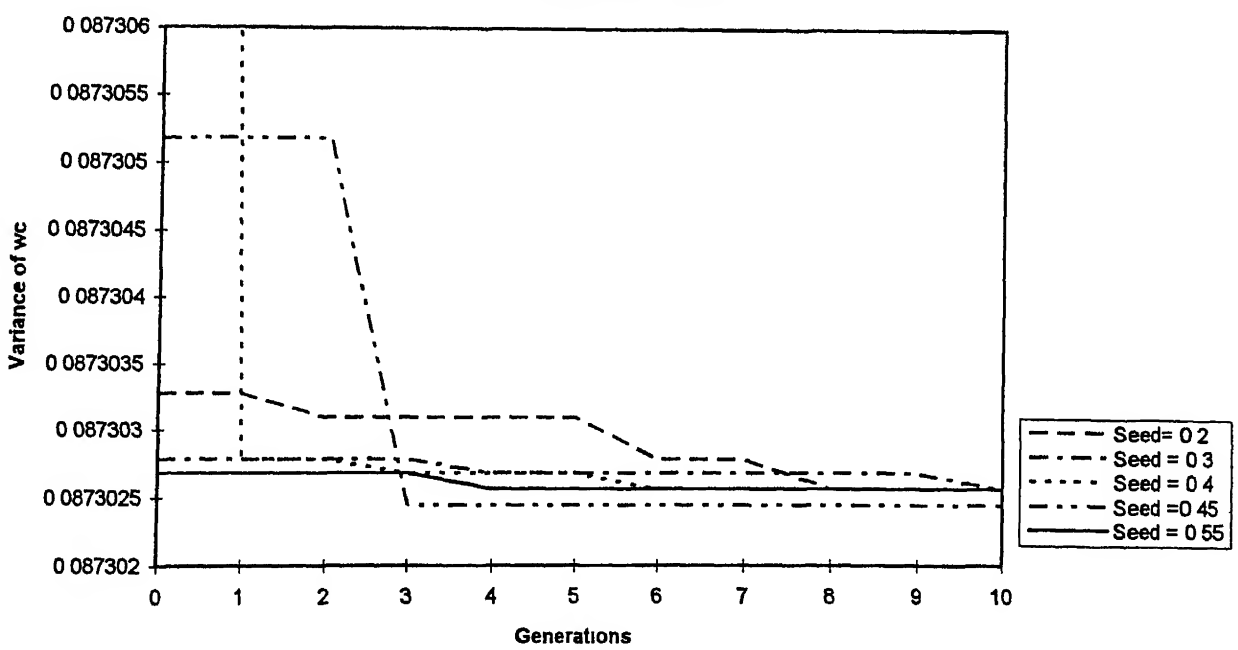


Table 3 14 Convergence of the variance towards the best individual

Generation	Seed= 0 2	Seed = 0 3	Seed = 0 4	Seed =0 45	Seed = 0 55
0	0 087303276	0 087302785	0 08733997	0 087305183	0 087302685
1	0 087303276	0 087302785	0 087302785	0 087305183	0 087302685
2	0 087303099	0 087302785	0 087302785	0 087305183	0 087302685
3	0 087303099	0 087302785	0 087302685	0 087302446	0 087302685
4	0 087303099	0 087302685	0 087302685	0 087302446	0 08730257
5	0 087303099	0 087302685	0 087302685	0 087302446	0 08730257
6	0 087302785	0 087302685	0 08730257	0 087302446	0 08730257
7	0 087302785	0 087302685	0 08730257	0 087302446	0 08730257
8	0 08730257	0 087302685	0 08730257	0 087302446	0 08730257
9	0 08730257	0 087302685	0 08730257	0 087302446	0 08730257
10	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
11	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
12	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
13	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
14	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
15	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
16	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
17	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
18	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
19	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
20	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
21	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
22	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
23	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
24	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257
25	0 08730257	0 08730257	0 08730257	0 087302446	0 08730257

Fig 3.7 The Convergence of Variance wc to the best solution with generations in experiment # 20



CHAPTER 4

PARETO OPTIMAL SOLUTIONS USING MULTIOBJECTIVE GA

The filter design problem has two objectives The cutoff frequency (ωc) must be robust, centered at 6 84 Hz, and the galvanometer deflection (D) must be centered at 3 00" and also be robust Such problems are multi-objective optimization problems In such problems, designer is interested in a *set* of solutions with Pareto optimal property or some such criteria, instead of a single solution Since the genetic algorithm (GA) works with a population of solutions, it would seem natural to use GA on multi-objective optimization problems to capture several solutions simultaneously that satisfy the Pareto-optimality criteria The first practical multi-objective genetic algorithm, called Vector Evaluated Genetic Algorithm (VEGA) was developed by Schaffer in 1984 [24] Schaffer modified the simple tripartite (using reproduction, crossover and mutation as the three main operators) genetic algorithm (SGA) by performing independent selection cycles according to each objective He modified Grefenstetts's GENESIS program [25] by creating a loop around the traditional GA procedure so that the selection method is repeated for each individual objective to fill up a portion of the mating pool Then the entire population is thoroughly shuffled to apply crossover and mutation operators This is performed to achieve the mating of individuals in different sub-population groups

VEGA works efficiently for some generations, but in some cases it suffers from its bias towards some individuals or regions The independent selection of specialists (solutions good in one objective) results in *speciation* in the population The outcome of this effect is the convergence of the entire population toward individual optimum regions after a large number of generations One method of minimizing speciation is through the nondominated sorting procedure in combined with a special fitness sharing technique, as suggested by Goldberg [26] One such algorithm that we used to solve

the filter design problem is different from VEGA It uses sharing and is known as the binary coded Nondominated sorting genetic algorithm (NSGA) [4]

4.1 Nondominated Sorting Genetic Algorithm (NSGA):[4]

The idea behind the nondominated sorting procedure is that a *ranking selection method* is used to emphasize good solutions and then a niche method is used to maintain stable subpopulation of good points Thus it differs from the SGA in the manner the selection operator works The crossover and mutation operators work as usual However, before the selection is performed, the population is ranked on the basis of a nondomination concept to achieve Pareto-optimality as follows [MCDM-Chapter1]

The nondominated individuals are first identified from the current population All these individuals are assumed to contribute the first nondominated front in the population and assigned a dummy fitness value proportional to population size This dummy fitness is intended to give equal reproductive chance to all these nondominated individuals The diversity in population is maintained by sharing among the individuals their fitness values (Fitness sharing is a way of finding and maintaining multiple optimal solutions in a simple GA-[4]) These individuals in the first front are then ignored temporarily in order to genetically process the rest of the population in a similar way Thus individuals for the second front are identified These Second-front solutions are then assigned a new dummy fitness value which is kept smaller than the minimum shared dummy fitness of the previous front This process is continued until the entire population is classified in to several successive fronts of "nondominated" individuals

To illustrate the nondominated sorting procedure we may use the following example [4] - Consider a population of Six individual solutions as shown in Table 4 1 The problem involves one decision variable x and two objectives x^2 and $(x - 2)^2$ to be minimized

Table 4.1

Individual	Design Variable(x)	Fitness-1 x^2	Fitness-2 $(x-2)^2$	Rank/ Front	dummy fitness before sharing	Dummy fitness after sharing
1	-1 5	2 25	12 25	2	3 00	3 00
2	0 7	0 49	1 69	1	6 00	6 00
3	4 2	17 64	4 84	2	3 00	3 00
4	2 0	4 0	0 00	1	6 00	3 43
5	1 75	3 06	0 062	1	6 00	3 43
6	-3 0	9 00	25 0	3	2 00	2 00

Each individual here has two fitness values The first fitness corresponds to the objective $f_1 = x^2$ while the second fitness relates to a second objective $f_2 = (x-2)^2$ Using concept of nondominance, these individuals may be ranked as follows

Since both fitness values, f_1 and f_2, of the second individual are less than that of first individual, second individual dominates the first individual Proceeding in this manner the first, third and sixth individuals are classified as dominated points

Thus the sorting will produce the first batch of nondominated points- second, fourth and fifth individuals- as they dominate each other either in f_1 or in f_2 These individuals would belong to the first front and each is initially assigned a high dummy fitness value, say 6 0 Once the first front has been found, all points in the first front are ignored temporarily, to process the rest of the population in the same way to develop the second nondominated front etc Thus we are left with first, third and sixth individuals to cinsider further Among these points, the first and third individuals get classified as the second front and sixth point becomes a lone member of the third front In each front, dummy fitness value is assigned in a way so that a later front is inferior to a former front in dummy fitness For example, to first and third individuals, a fitness of 3 00 each and to sixth a fitness of 2 00 may be assigned In actual practice, the dummy fitness values assigned would depend also on the sharing of fitness among

the individuals on a particular front that is to occur, an aspect discussed in the following section The rule is that these newly assigned values are kept slightly *less* than the minimum shared dummy fitness of the previous front This method helps to set priorities on the different individuals in the total population

Next the individuals in the population are reproduced according to their relative dummy fitness values A stochastic remainder proportionate [4] selection method has been used in this study In this method individuals in the first front having the maximum fitness value get more copies made up of them than rest of the population This approach facilitates the search for nondominated regions or Pareto-optimal fronts This results in quick convergence of the population towards the nondominated region while sharing helps to distribute the individuals over this region By emphasizing the nondominated points, NSGA favors the schemata representing Pareto-optimal region

The efficiency of NSGA lies in the way multiple objectives are reduced to a dummy fitness function using a nondominated sorting procedure If a point is locally dominated, it is also globally dominated However, the converse is not true [4] In each generation NSGA finds locally nondominated points only However if there exists any globally nondominated member, NSGA insures its survival This is because any such individual is likely to have the highest possible dummy fitness Thus the power of NSGA lies in the successful transformation of a multiobjective problem, no matter how many objectives are there, to a single function problem, without losing the concept of vector optimization By selecting nondominated points, NSGA actually processes those schemata that represent Pareto-Optimal regions Therefore, the building blocks (in the sense Goldberg uses this phrase) for NSGA will be those schemata that represent globally nondominated individuals

The Figure 4 1 shows the flow chart of the NSGA algorithm Both maximization and minimization problems can be solved by this algorithm by suitably altering the selection process of nondominated individuals For instance, if the

objective is to maximize functions we define a dominated point if the corresponding objective component is *not greater than* that of a nondominated point

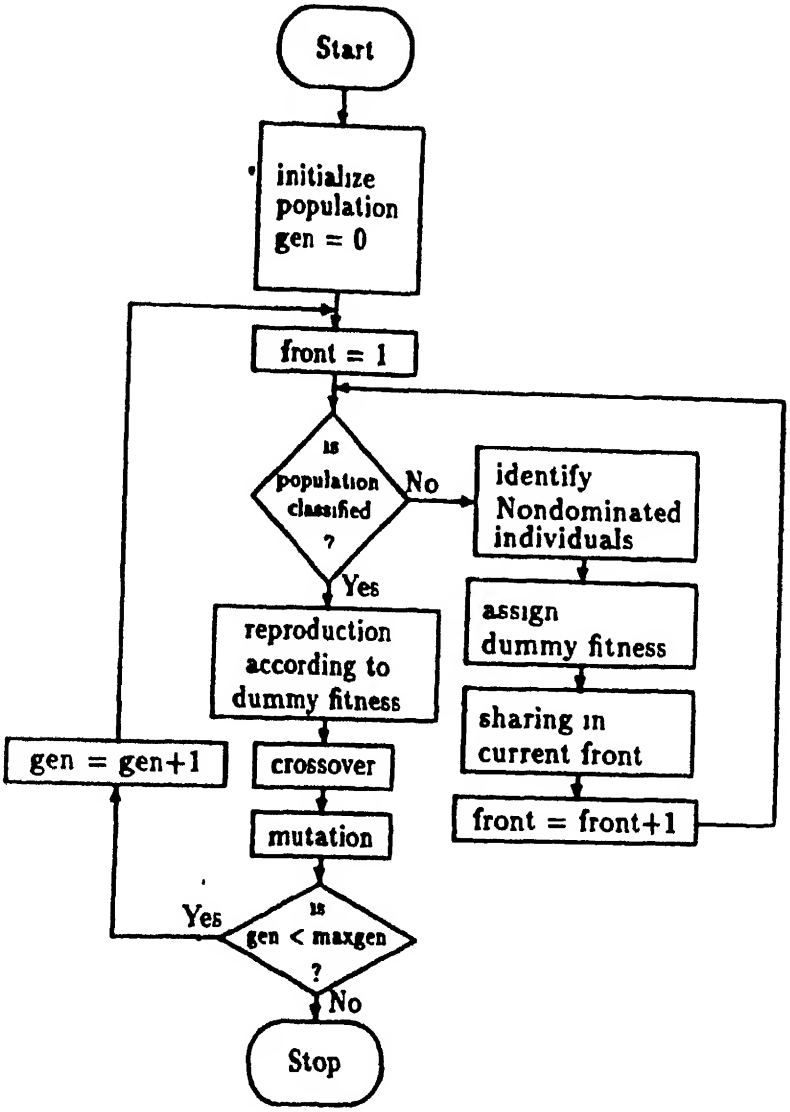


Fig 4.1 NSGA flowchart [4]

4.2 Sharing of dummy fitness among members on a front [4]

The NSGA algorithm operates in a manner that is similar to the simple GA except for the manner of *classification* of the members into successive nondominated fronts and the subsequent *sharing* operation The sharing of fitness within each front is achieved by calculating a sharing function value $Sh(d_{ij})$ between two individuals i and j in the same front as follows

$$Sh(d_{ij}) = 1 - (d_{ij} / \sigma share), \quad \text{if } d_{ij} < \sigma share,$$
$$= 0, \quad \text{otherwise}$$

In the above definition of $Sh(d_{ij})$, the parameter d_{ij} is the *phenotypic distance* allowed between i and j in the current front, and σ_{share} is the maximum phenotypic distance allowed between any two individuals to become members of a *niche* The phenotypic distance (d_{ij}) is measured based on the differences in the decoded problem parameters (the values of x in Table 4 1) There may be another way of evaluating the sharing functions using the number of different bits between two individuals Such sharing is called genotype sharing A niche is a stable, noncompeting subpopulation of the points surrounding the important peaks in the search space A parameter *niche count* is calculated adding the sharing function values for all the individuals in the current front Finally, the *shared* fitness value of each individual is calculated by dividing its dummy fitness value by its niche count

There are variations to the this procedure, as in MOGA by Fonesco and Fleming[27] In MOGA the whole population is checked and all nondominated individuals are assigned rank 1 The other individuals are ranked by checking the nondominance of them with respect to the rest of the population, as follows

First, for an individual member, the number of members in the population that strictly dominate this member are found Thereafter, the rank of that individual is assigned to be more than one that number Therefore, at the end of this ranking procedure, there could be a number of points having the same rank The selection procedure then uses these ranks to select or delete blocks of points to form the mating pool However as discussed by Goldberg and Deb [23], this type of blocked fitness assignment is likely to produce a large selection pressure that might cause premature convergence

MOGA also uses a niche-formation method to distribute the population over the Pareto-optimal region But instead of performing sharing based on the problem parameter values as in NSGA, MOGA uses sharing on objective function values The important aspect of this work is the ranking of the individuals according to their

nondominance in the population Horn, Nafpliotis and Goldberg[34] used Pareto domination *tournaments* instead of the sorting and ranking selection method in solving multiobjective optimization problems NSGA [4] uses both aspects of Goldberg's suggestions namely, a ranking selection method is used to emphasize good points and niche forming method used to maintain stable subpopulation This improves the process of seeking the Pareto-optimal solutions

4.3 The Primary NSGA subroutines and their functions as implemented by Deb and Shrinivas:-

Main.c:- This is controller part of the program that calls the different subroutines, each of which performing a specific task as follows

(i)Initialize:- This subroutine initializes the different parts of the program First the *initdata* subroutine obtains the basic problem values from an input file called nsga in *initmalloc* allocates space for global data structures *app_init* initializes the application dependent variables *randomize* initializes the random number generator *initpop* initializes the population and reports statistics *initdata* does data inquiry and setup *initreport* asks for the different GA parameters to be used by NSGA during its execution

(ii)Application:- This subroutine contains the actual problem or the function to be optimized In this routine a subroutine *phenoshare* is called Here oldpopulation is multiplied with dummy fitness and the product is divided by niche count The routine *distance* returns the phenotypic distance between two individuals in the n-dimensional multiobjective space

(iii)fronts - This is perhaps the most important routine of the NSGA implementation Here the entire population is classified into different fronts and dummy fitness is assigned and shared within each front The population is evaluated for each objective and the individuals are classified according to their original fitness value into different fronts Thus each individual in front 1 will have better fitness value at least for one objective while for other objective at least not smaller than any of the individuals in

the front 2 Same is between front 2 and 3, and so on until the complete population is classified into a number of fronts

(iv)Generate:- This routine is same as the routine *generation* in SGA [Chapter 2] Here a preselection action is performed before the generation Then the genetic operators, crossover and mutation are applied on the population by calling in the *operators* routine

All the other supporting routines in NSGA are identified to the corresponding SGA routines except for memory management routines and the Pareto-report routine The following is the list of these supporting subroutines -

(v) Random:- Random number generator,

(vi) Initial:- Initialization and setup of functions and routines are done in this routine

(vii) Statistics:- Computes the maximum, minimum and the average of the fitnesses (simple and dummy used by NSGA)

(viii) Report:- This routine prints the generation statistics report as well as the population report

(ix) Result:- This routine stores the results of one particular run of the NSGA algorithm

(x) Preselect:- Preselection for the stochastic remainder selection method are done in this routine

4.4 THE CONCEPT OF SHARING:- (A GA with sharing incorporated for multimodal optimization developed by Goldberg and Richardson [29])

Simple GAs have been criticized as having sub-par performance on multimodal functions They have also been criticized for the so-called premature convergence where substantial fixation occurs at most bit positions before obtaining sufficiently near-optimal points [27] To overcome the first of these shortcomings a method from nature has been borrowed This remedy of *sharing functions* helps mitigate unbridled head-to-head competition between widely disparate points in a search space As a side

advantage we find that sharing helps maintain a more diverse population and a more "considered" convergence

It has been found that the SGA, when operating on multimodal functions clusters all of its points about one peak or the other The reason for this is that the fundamental theorem of GA assumes an infinitely large population size In finite size population even when no advantage exists for either of two competing alternatives, the population will converge to one or the other of many in finite time This phenomenon is called *genetic drift* However, in real situations we are often interested in finding good, better and best solutions This is similar to multimodal peaks of unequal size In this case it would be desirable to see a form of convergence that permits stable sub-population of points to cluster around each peak according to peak fitness

In order to overcome these deficiencies of the SGA, the concept of niche and species have been borrowed from nature where different species don't always compete head-to-head Instead they exploit separate sets of environmental features (niches) in which other organisms have no interest A number of schemes have been implemented to induce niche and species developement concepts in genetic algorithms

Cavicchio [28] in his dissertation introduced a mechanism called *Preselection* In this scheme, an offspring replaces the *inferior* parent if the offspring's fitness exceeds that of the inferior parent In this manner diversity is maintained in the population because strings tend to replace strings *similar* to themselves Using Preselection, Cavicchio claimed to maintain a more diverse population in a number of simulations with relatively small population sizes (n=20) Dejong [9] proposed crowding scheme as an alternative In Dejong's crowding, individuals replace existing strings according to their similarity with other strings in an overlapping population Dejong demonstrated good success with the crowding scheme on multimodal functions Schaffer [24] used separate, fixed size subpopulation in his study of the vector evaluated genetic algorithm (VEGA) This method works well for a number of trial functions, however, Schaffer has expressed concern over the procedure's ability to

handle middling nondominated individuals- individuals that may be Pareto optimal but are not extremal along any single dimension

Perry's [29] work involves a genotype-to-phenotype mapping, a multiple-resource environment, and an special entity called an external schema External schemata are special "similarity templates" defined by the designer to characterize species membership Unfortunately, the required intervention of an outside agent limits the practical use of this technique Mauldin [29] has attempted to maintain diversity in genetic algorithm through his uniqueness operator, without directly addressing the niche and species

In the next section we show how we can maintain appropriate diversity through the use of sharing functions

4.4.1 SHARING FUNCTIONS [29]

In attempting to induce species formation in the population we are faced with two questions who should share fitness and how much fitness should be shared? We answer these question by introducing the sharing function A sharing function is a way of determining the degradation of an individual's payoff due to a neighbor at some distance as measured in some similarity space Mathematically, we introduce a convenient metric dij over the decoded parameters x_i and x_j(the decoded parameters are themselves functions of the strings si because $x_i = x_i(s_i)$)

$$d_{ij} = d(x_i, x_j),$$

Alternatively, one may introduce a metric over the strings directly

$$d_{ij} = d(s_i, s_j)$$

We will use a metric defined over the decoded parameters xi (phenotypic sharing) for the following discussion A sharing function sh is defined as a function of metric values (d) as sh = sh(d), with following three properties

1 $0 \le sh(d) \le 1$, for all d

2 $sh(0) = 1$

3 $\lim_{d \to \infty} sh(d) = 0$

Of many possible sharing functions, The power law function is convenient

$$Sh(d_{ij}) = 1 - (dij / \sigma share), \quad \text{if } dij < \sigma share,$$

$$= 0, \quad \text{otherwise}$$

Once we have selected a metric and a sharing function, it is a simple matter to determine a string's shared fitness We define a string's shared fitness f' as its potential fitness divided by its niche count mi'

$$f_i' = f_i / m_i'$$

The niche count m_i' is taken as the sum of all share function values taken over the entire population of individuals

$$m_i' = \sum_{j=1}^{N} sh(dij) = \sum_{j=1}^{N} sh(d(x_i, x_j))$$

Since the sum includes the string itself, if a string is all by itself in its own niche ($m_i' = 1$), it receives as its shared fitness its full potential fitness value Otherwise the sharing function degrades fitness according to the number and the closeness of the neighboring points As an alternative, sharing functions may be evaluated using the genotype (the string) directly

The degree of sharing may be decreased to reduce the level of computations by taking a random sample of k ($k \ll N$) share function values and extrapolating the mean If the mean of k shared function evaluations for string i is μ_i and the population is of size N, then the following formula provides a reasonable way to estimate the niche count m_i'

$$m_i' = (N-1)\mu_i + 1$$

Goldberg and Richerdson tested the concept of sharing on two test functions f1(x) and f2(x) shown below - The first function is a periodic function with five peaks of equal magnitude [29]

$$f1(x) = \sin^6(5.1\pi x + 0.5)$$

Probability of mutation Pm = 0.0

Probability of crossover Pc= 0.8

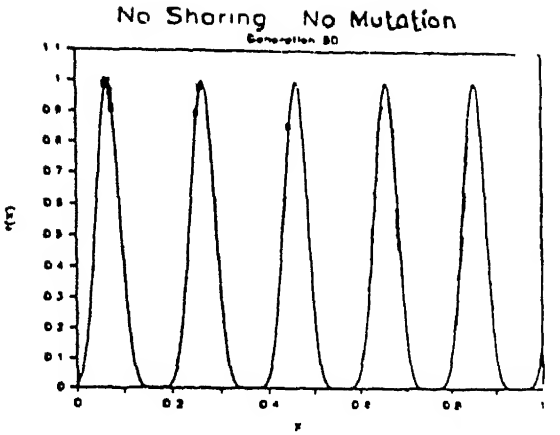
Population size Ps = 50

Maximum no of generation = 100

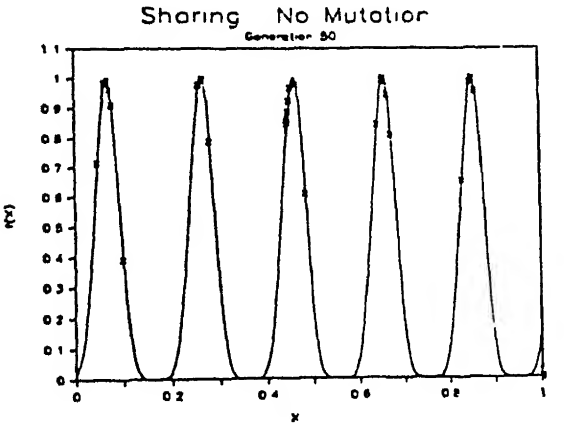
The second test function has five peaks of decreasing magnitude and is given by the equation

$$f2(x) = f1(x)\ e^{[-4\ln 2((x-0.0667)^2/0.8^2)]}$$

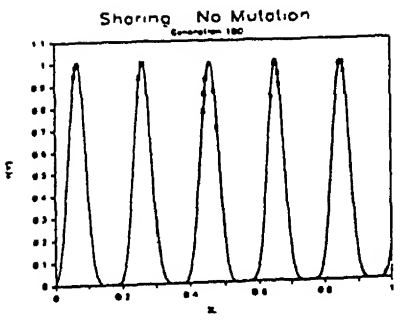
The graphs in Fig 4.2 and 4.3 obtained from [29] show the comparison of the performance of the simple genetic algorithm with no sharing and no mutation to the performance of the same GA with sharing. The parameter were held constant across all runs. For both the functions, the GA with sharing was able to maintain stable sub-population about significant peaks while an identical GA without sharing was unable to maintain points at more than a single peak. Figures 4.2 and 4.3 illustrate how the GA with sharing distributes points to all peaks



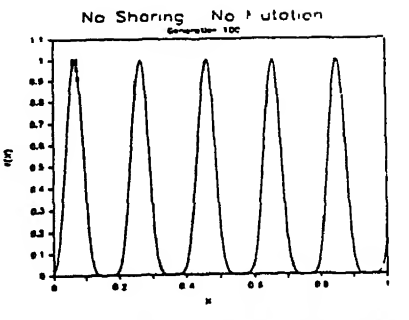
GA without sharing concentrates points at three peaks after 50 generations on function f_1 (equal peaks)



GA with sharing distributes points to all five peaks after 50 generations on function f_1 (equal peaks)

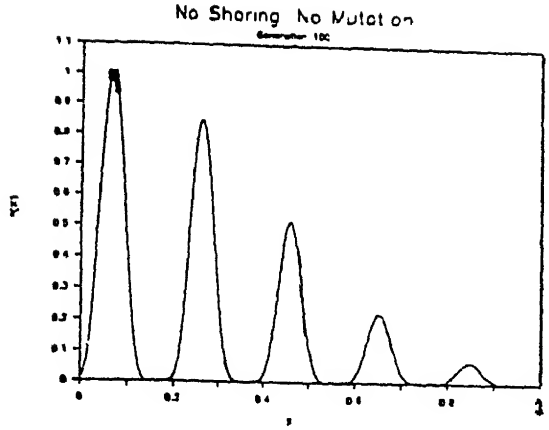


GA with sharing continues to distribute points among all five peaks after 100 generations on function f_1 (equal peaks)

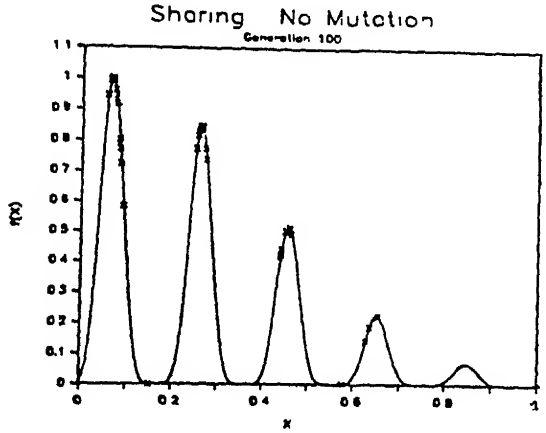


GA without sharing concentrates all points on a single peak after 100 generations on function f_1 (equal peaks)

Fig 4 2 GA with and without sharing for function f1



GA without sharing concentrates all points on highest peak (at generation 100) on function f_2 (decreasing peaks)



GA with sharing distributes points to all but lowest peak (at generation 100) on function f_2 (decreasing peaks) Lowest peak has expected population size of only one member, not enough to overcome selection and sampling errors

Fig 4 3 GA with and without sharing for function f2

CHAPTER 5

APPLICATION OF NSGA IN FILTER DESIGN PROBLEM

As we know, there are two objectives in the robust filter design problem minimize the variance of wc and minimize the variance of D The use of SGA to find the best solution requires the application of method of objective weighting, where the number of objectives are combined in to a single objective by assigning each individual objective a suitable weight The solutions obtained depends upon the weights to be assigned to the individual objectives This is a problem in itself We are using here NSGA algorithm which uses the niche and speciation method to find the multiple Pareto-optimal points with the concept of sharing to find solutions for multimodel functions This is possible without going through the tedious process of knowing what weights should be assigned for individual objectives

The results of Parameter optimization experiments done in Chapter 3 are utilized here The best parameters identified in Chapter 3 were -

Population size Ps = 100,

Crossover probability Pc = 0 75,

Mutation probability Pm = 0 05,

The following Table 5 1 shows how the number and quality of first rank solutions (Pareto-optimal solutions) (Chapter 1) change after each ten successive generations in NSGA for these parameter settings The averages are taken for the number of first rank solutions in each intervals of 10 generations

Table 5.1

Generation #	No of first rank solutions	Average Var(wc)	Average Var(d)
1	66	0 090309	0 010893
10	90	0 091912	0 010793
20	93	0 090891	0 010825
30	93	0 090123	0 010848
40	89	0 089852	0 010908
50	91	0 090536	0 010803

Figures 5 1-5 6 show the Pareto-optimal fronts of first rank solutions generated after each ten successive generations Any one of these solutions is a Pareto-optimal solution The choice of a particular solution would now depend upon the designer's preference and availability of the appropriate value of components in the market We have selected here 25 first rank solutions at random (Table-5 3) from generation # 50 By carefully observing the results we find that if a solution in front 1 is better than any other solution with respect to one objective {say Var(wc)} than it worse with respect to the other objective {Var(d)} or vice-versa This is the property of nondominance

Now we compare the results of the Monte-carlo simulation approach used by Bagchi and Kumar and the constrained optimization approach Table 5 2 shows the two extreme members of the solutions obtained by Bagchi and Kumar They could identify only those extreme solutions which are robust with respect to only one criterion but not so robust with respect to the other They further said that by search and enumeration one may find any acceptable solution between these two solutions The use of NSGA does exactly that for us in the Constrained optimization approach The solutions identified by Bagchi and Kumar were -

R3 = 300Ω , R2 = 29Ω and C = 454 4μf

(this design maximizes the S/Nwc or robustness with respect to wc),

R3 = 200Ω, R2 = 106Ω and C = 424 1μf

(this design maximizes the S/Nd or robustness with respect to D)

Table 5.2

R3(ohm)	R2(ohm)	C(μ f)	Avg(wc) (Hz)	Var(wc)	Avg(d)(in)	Var(d)
300	29	454 4	6 8458	0 088267	3 006	0 01108
200	106	424 1	6 8466	0 092578	3 008	0 01054

Table 5.3

R3(ohm)	R2(ohm)	C(farad)	Avg(wc)(hz)	Var(wc)	Avg(d)(in)	Var(d)
311 7881	22 43919	0.000462	6 841476	0 087877	3 00349	0 011188
156 2765	157 7785	0 000434	6 844165	0 095864	3 003232	0 010474
306 0652	25 66875	0 000458	6 841552	0 088029	3 003469	0 011125
195 026	111 5562	0 000424	6 8434	0 092806	3 00325	0 010506
225 7759	82 24873	0 000426	6 842843	0 091095	3 00328	0 010581
195 1129	111 4653	0 000424	6 843399	0 092801	3 00325	0 010507
299 6366	29 40162	0 000454	6 841641	0 088213	3 003447	0 011058
269 8776	48 28555	0 000439	6 842093	0 089215	3 003361	0 010806
292 3232	33.78978	0 00045	6 841746	0 088438	3 003423	0 010988
341 029	7 189255	0 000483	6 841166	0 087348	3 003618	0 011571
197.1142	109.3861	0 000424	6.843362	0 092675	3 003252	0 01051
225 9195	82 1241	0 000426	6 84284	0 091088	3 003279	0 010581
147 579	169.993	0 000438	6 84435	0 096766	3 003229	0 010473
324 7787	15.4167	0 000471	6 841321	0 087583	3 003543	0 011346
314 7266	20 81395	0 000464	6 84144	0 087804	3 003502	0 011222
337 4986	8 926979	0 00048	6 841196	0 087384	3 003601	0 01152
204.7699	101 6687	0 000424	6 84322	0 092214	3 003258	0 010524
229 909	78 70221	0 000427	6 842768	0 090896	3 003285	0 010595
303 139	27 35382	0 000456	6 841593	0 088111	3 003458	0 011094
242 519	68 38161	0 00043	6 84255	0 09032	3 003304	0 010647
340 9459	7 229851	0 000483	6 841167	0 087349	3 003618	0 01157
270 2351	48 0416	0 000439	6 842086	0 089201	3 003361	0 010808
242 6987	68 23972	0 00043	6 842547	0 090312	3 003305	0 010648
165 1582	146 0743	0 00043	6 843982	0 095038	3 003235	0 010478
202 6415	103 7775	0 000424	6 843259	0 092339	3 003255	0 01052

The Table 5 2 shows the two solutions obtained by Bagchi and Kumar [13] and Table 5 3 shows the 25 first rank solutions obtained using NSGA The parameter settings

are as above mentioned As said above these 25 results are taken at random from the generations # 50 Though it is not very clear by comparing the variances, which of the two methods gives better results since we find that the variances for the two FRs appears to be comparable in the two methods, but if we carefully look at the average values of wc and D in Tables 5 2 and 5 3, we find that the *order* of the deviations of the means from the target values, wc = 6 84 Hz and D = 3 00 in is considerably less in the solutions obtained by the COP method This is illustrated in the following Tables 5 4

Table 5.4

Solution by	R3 (ohm)	Deviation of Mean wc from target.	Deviation of mean D from target.
Bagchi [13] using Monte-Carlo simulations	200 300	0066 0058	008 006
COP Using NSGA.			
Sol.#-1	311.7881	.001476	00349
2	156.2765	004165	003232
3	306.0652	001552	003469
4	195.026	003400	00325
5	225.7759	002843	00328
6	195.1129	003399	00325
7	299.6366	001641	003447
8	269.8776	002093	003423
9	292.3232	001746	003423
10	341.029	001166	003618
11	197.1142	003362	003252
12	225.9195	00284	003279
13	147.579	00435	003229
14	324.7787	001321	003543
15	314.7266	00144	003502
16	337.4986	001196	003601
17	204.7699	00322	003258
18	229.909	002768	003285
19	303.139	001593	003458
20	242.519	00255	003304
21	340.9459	001167	003618
22	270.2351	0002086	003361
23	242.6987	002547	003305
24	165.1582	003982	003235
25	202.6415	003259	003255

This Table indicates that the mean values of wc and D for the solutions obtained by COP using NSGA remains more close to the target values, wc = 6 84 Hz and D = 3 00, in the presence of noise as compared to those obtained using Monte-Carlo approach whereas the variance of wc and D are as robust as in case of Monte-Carlo approach (Table-5 3) This illustrates the advantage of COP approach over the two earlier methods employed by Filippone [30] and Bagchi [13][15] We have not used here the results of the Taguchi's two step method employed by Filippone for the comparison purpose because as we noted in the Chapter 3 that the solutions obtained by Filippone were not even fulfilling the target requirements The reason for the solutions obtained by Filippone being off target is that he had made an assumptions that the individual factor effects are not interacting, Which is not true as we observed in the graphs 3 2 1-3 2 12, which show complex interactions between the three design parameters This exercise illustrates first the limitations of the Taguchi's two step method in robust design problems where more than one functional requirements are to be made robust and there are strong interactions are present between design parameters as in case of the filter design problem Second is the demonstration of improvements possible over the solutions obtained by Bagchi and Kumar using Monte-Carlo approach [13]

Fig 5.1 The Pareto-optimal front at Generation 1
66 members.

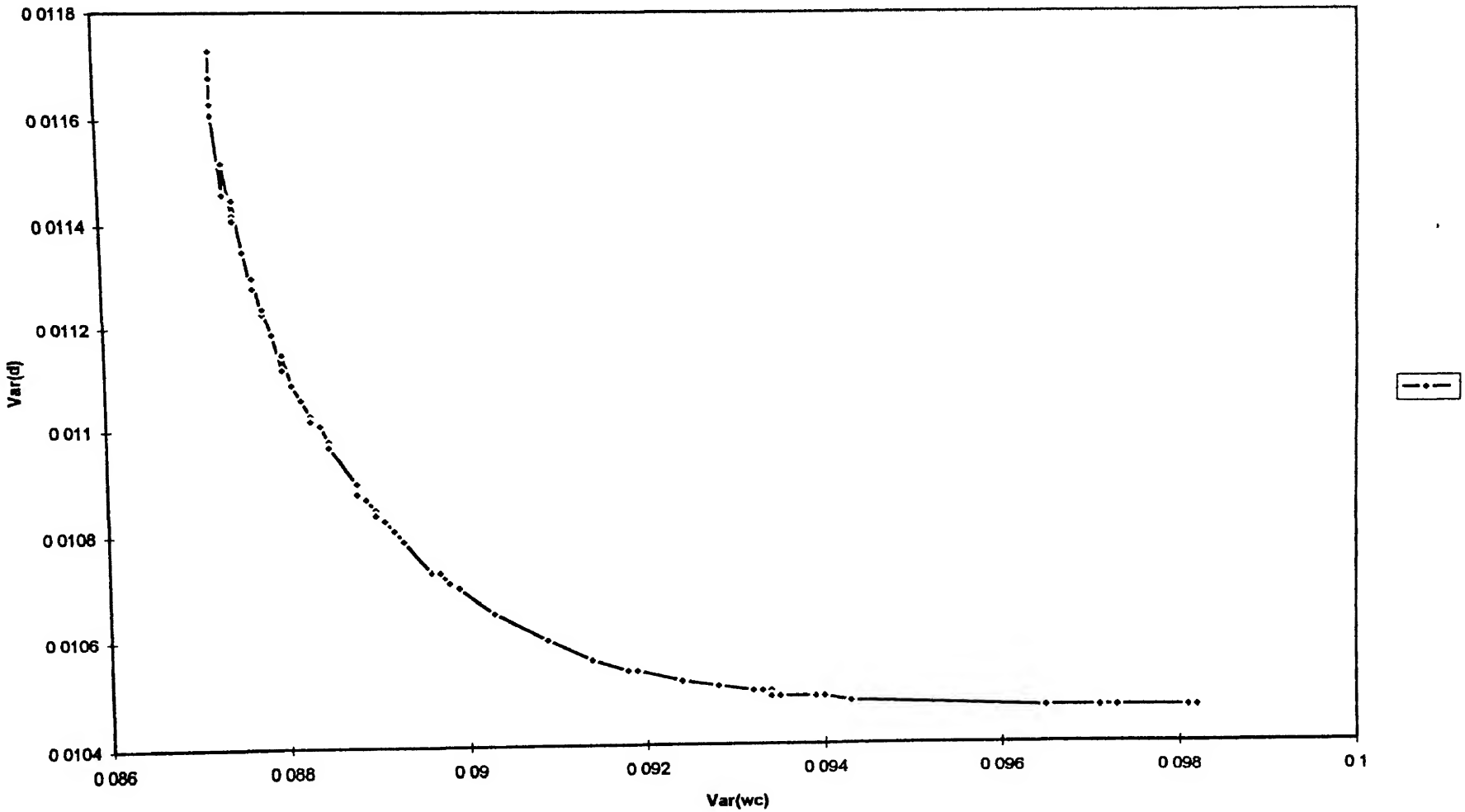


Fig 5.2 The Pareto-optimal front at Generation 10
90 members.

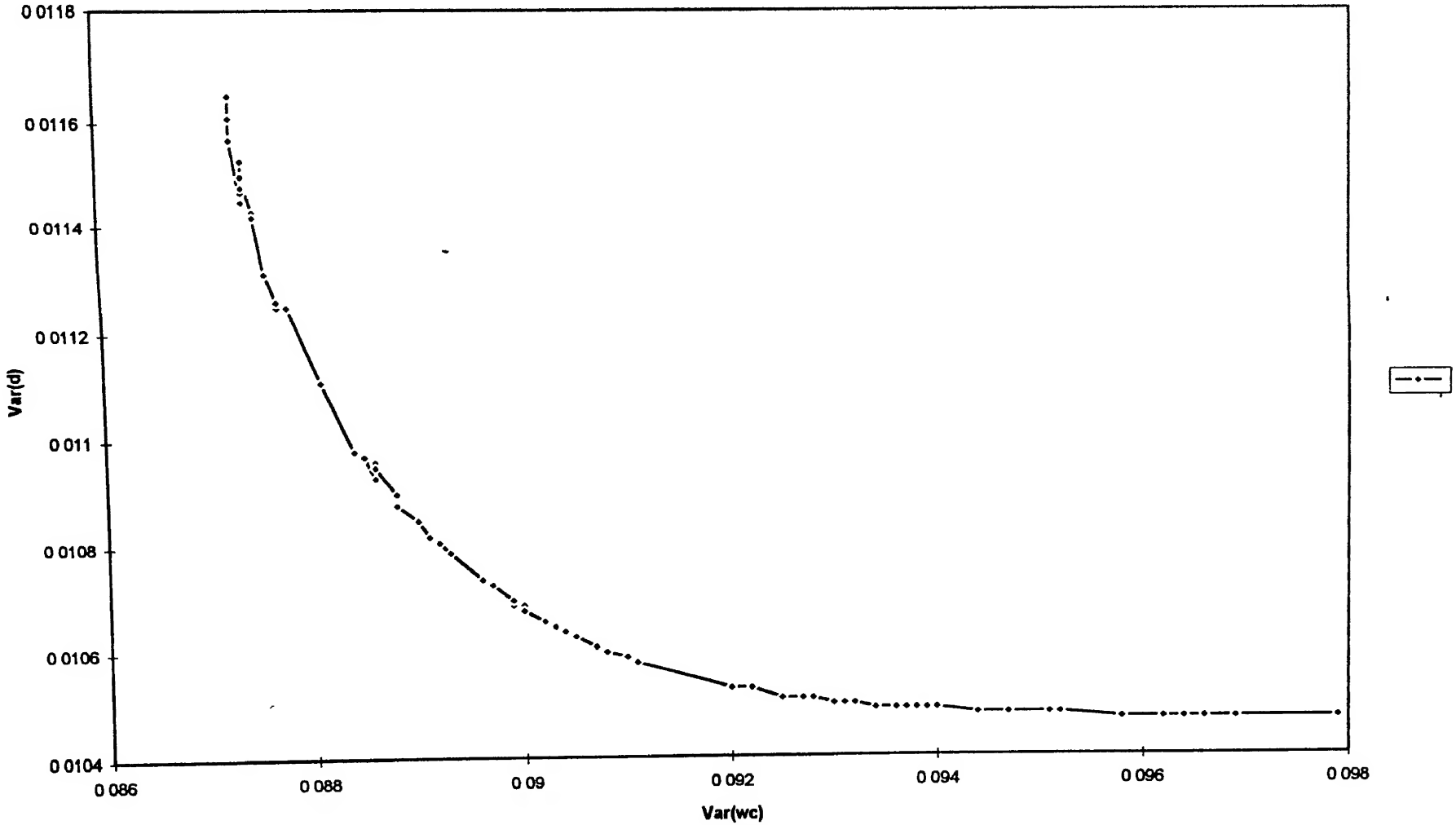


Fig 5 3 The Pareto-optimal front at generation 20
93 members.

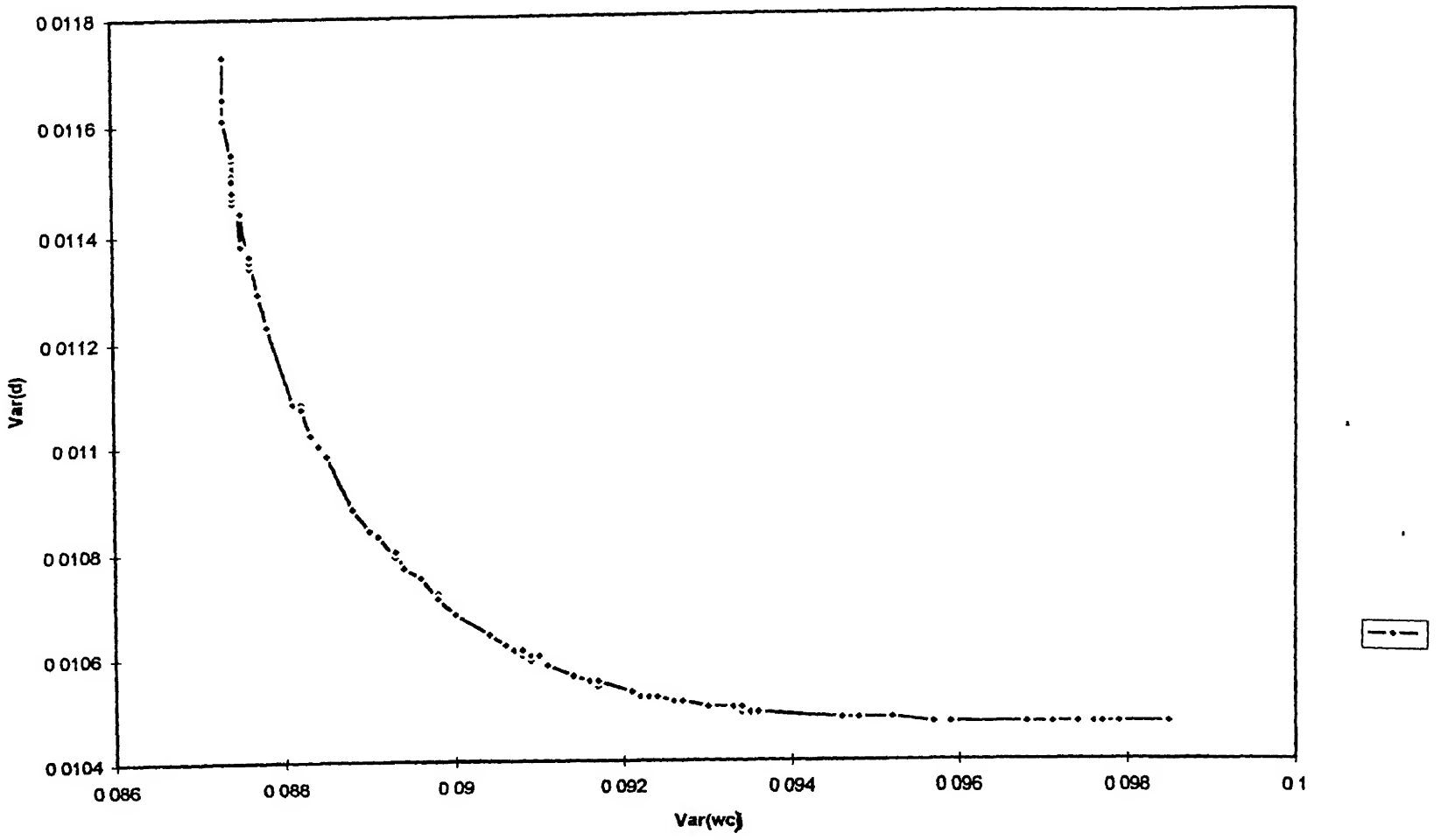


Fig 5.4 The Pareto-optimal front at generation 30
93 members

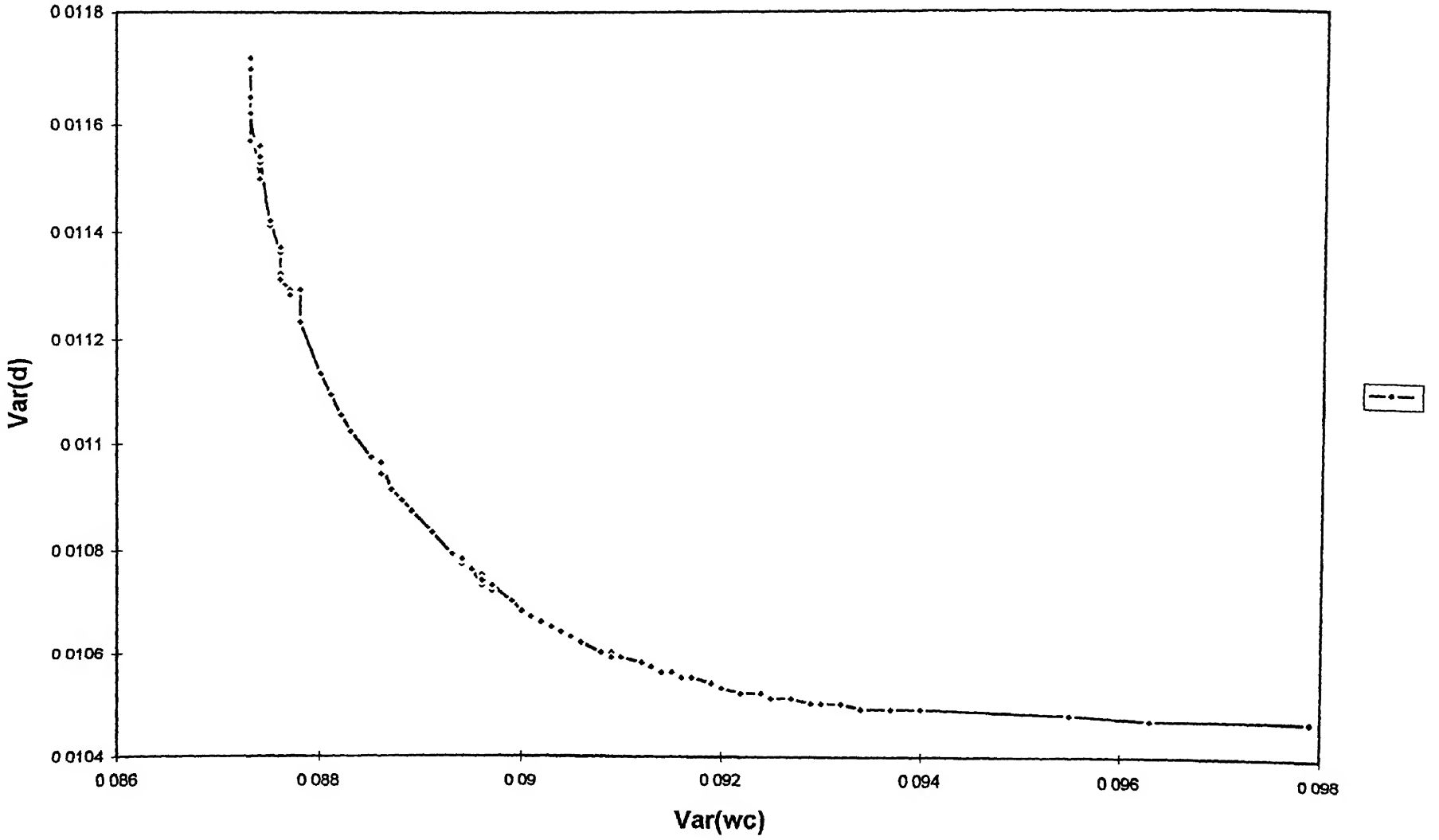


Fig 5 5 The Pareto-optimal front at generation 40
89 members

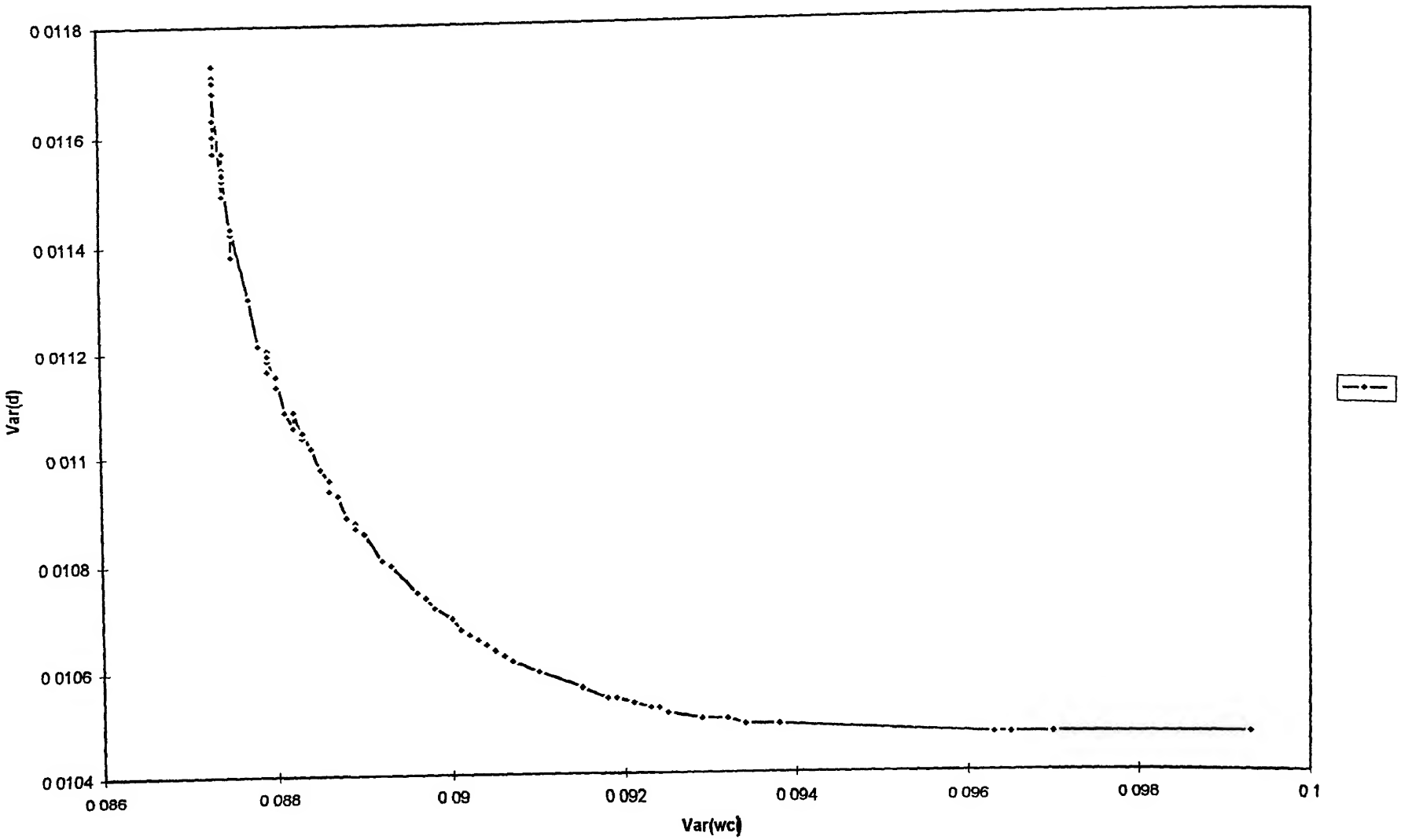
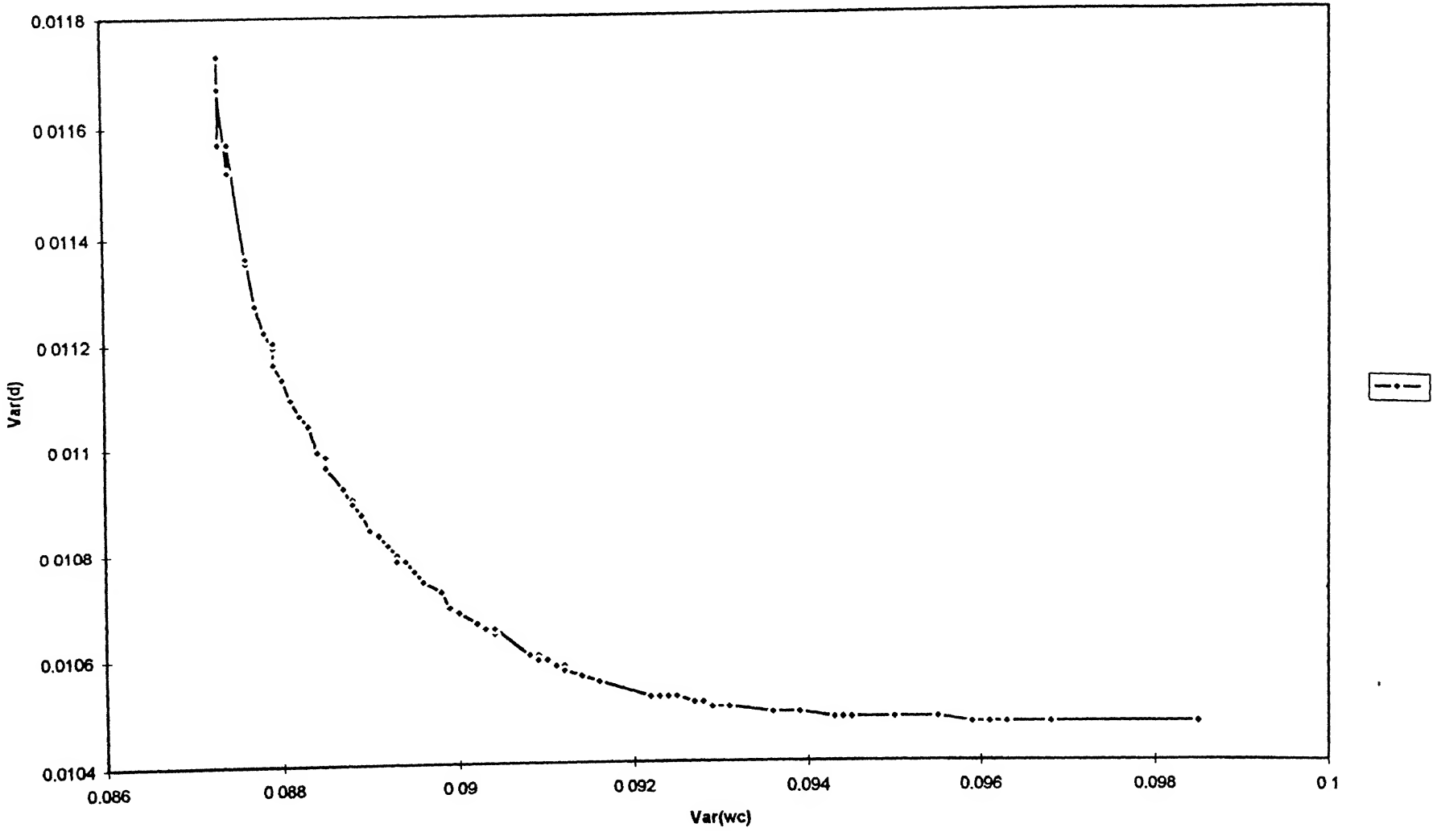


Fig 5.6 The Pareto-optimal front at generation 50
91 members.



CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In Chapter 1 we reviewed the robust design philosophy. The need for robust design The impact of the off target design as a loss to the society. Taguchi's two step methodology to robust design The steps in the robust designs The guidelines for implementing the Taguchi methods. Some of the robust design problems and the a mention of the Passive filter design problem on which we concentrated our research work

Chapter 2 gave a brief introduction of the Genetic-algorithms. Which are fast emerging as efficient search and optimization algorithms. The features and structure of the GA are mentioned The SGA (simple genetic algorithm) that we used in the later chapters is studied in detail The short comings of the traditional approaches are discussed.

Chapter-3 discussed the filter design problem in detail. The results of the past work done on this problem The reasons for limitations of the earlier approach are exposed in this chapter. A new approach to the robust design where Taguchi's two step method fails is proposed. An parametric experimental study is done for the selection of the best GA parameters and the best parameter settings are identified.

Chapter 4 discussed the current approaches for the solutions of the multi-objective optimization problems The concept of Pareto-optimality is discussed The earlier Genetic algorithms for multi-objective optimization are briefly reviewed. The sharing and niche methods are looked at Concept of nondominance is mentioned and discussion of NSGA algorithm [4] is done The NSGA is later used in the filter design optimization problem.

Chapter 5 compared the results of the COP using NSGA with earlier results and established the superiority of COP using NSGA over the methods used to date.

To summarize the following conclusions may be drawn:-

(i) Taguchi's two step method cannot be applied in all problems In fact only a limited class of robust design problems can be solved using Taguchi's two step method. where the

functions requirements are not more than one and the design factor effects are not interacting.

(ii) It is possible to locate the globally optimum robust design while the search is restricted only to those designs which ensures the targets

(iii) GA based search and optimization technique can be applied efficiently to robust design problems to produce good results GA gives best performance for certain combinations of parameters only The selection of optimum parameters increases the efficiency of GA. Therefore, a systematic procedure like *design of experiments* should be applied for finding appropriate GA parameters

(iii) Many problems like problems in electronics circuit manufacturing where usually more than one performance requirements are to be made robust simultaneously, the Taguchi method fails to give globally optimum solutions

(iv) In such situations COP methods gives better results. This is more efficient and less expensive computationally than Monte-Carlo simulations

(v) NSGA gives Pareto-optimal solutions which are more robust with respect to the desired FRs Importantly, NSGA also gives multiple Pareto-optimal solutions, therefore designer is at liberty to choose a particular solution according to his preference. He also has a choice of changing his decision at a later point of time due to changes in the conditions that prohibits the implementation of a particular solution.

The solutions obtained using NSGA are free from many of the drawbacks of the traditional approaches The current results were obtained using the binary coded NSGA by N Shrinivas[4]. We expect a further improvement in the quality of the solutions with the real coded NSGA [33] A software could be developed to do robust circuit designs and interfaced with CAD tools and other tools such as SPICE The cluster of points obtained in the Pareto-optimal fronts Fig 5.1-5 6 are due to the fitness sharing. This helps distribute the population over all the possible global optimums. This is very useful property for multimodal functions The NSGA can also be used in robust design problems of more than two objectives

REFRENCES

[1] Phadke, M.S, (1989), " Quality Engineering Using Robust Design", Prentice Hall, New Jersey.

[2] Bagchi, Tapan P, (1993), " Taguchi Methods Explained - Practical Steps to Robust Design", Prentice Hall of India.

[3] Suh, N.P , (1990), " The Principles of Design", Oxford University Press, New York

[4] K Deb and N Shrinivas, " Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". 1995 Massachusetts Institute of Technology. Evaluationary Computations 2(3)· 221-248

[5] Zeleny, " Multi Criteria Decision Making", Fumiko Seo and Masatoshi Sakawa, " Multi Criteria Decision Analysis".

[6] K Raman, (1994). " An Evaluation of Genetic Algorithms for Robust design" M.Tech Thesis, Department of Industrial and Management Engineering, IIT Kanpur.

[7] Goldberg, D.E " Genetic Algorithms in Search Optimization and Machine Learning", Addison Wesley, Massachusetts.

[8] Tiwari, A. (1995). "Optimum Job Sequencing in a Flexible Manufacturing Cell Environment using GA" M Tech Thesis, IME IITK.

[9] De Jong, K.A (1975). " An analysis of the behaviour of a class of genetic adaptive systems Dissertation Abstracts International , 36(10).

[10] Baker, J. E (1885). " Adaptive selection methods for genetic algorithms". Proceedings of an International Conference on genetic Algorithms and Their Applications (pp 101-111) Pittsburgh · Carnegie-Mellon University.

[11] Barker, J. E. (1989). " How GAs work. A critical look at implicit parallelism." proceedings of the International Conference on Genetic Algorithm . 59-68

[12] Goldberg, D. E and Deb. (1991) " A comparative analysis of Selection schemes used in Genetic Algorithms " Dept of General Engg. at University of Illinois at ulbama

[13] Bagchi, T P and Kumar. " Multi-criteria robust design of electronic devices. IIT Kanpur. Journal of Electronics Manufacturing (1993) 3, 31-38.

[14] Phadke, M. S. (1982). " Quality Engineering Using Design of Experiments." Proceedings of the section on statistical Education, American Statistical Education. 11-20.

[15] Bagchi, T P and Templeton, J. G. C " Multi-criteria robust design using constrained optimization". Journal of Design and Manufacturing (1994) 4, 21-30.

[16] Rao, S. S " Optimization theory and applications, Willey Eastern Limited, New Delhi.

[17] Maghsodloo, S. (1990) " The Exact relation of Signal-to-Noise ratio to the Quality loss function " Journal of quality Technology. Vol.22(1): 59-67.

[18] Nair, V N and Pregibon, D. (1986). " A Data Analysis Strategy for Quality Engineering Experiments." Quality Controal, Robust Design and Taguchi Methods: Article 14: 289-265.

[19] Bagchi, T. P and Kumar. " Multi-criteria robust design of electronic devices. IIT Kanpur Journal of Electronics Manufacturing (1993) 3, 31-38.

[20] Filippone, S F " Using Taguchi Methods to apply to the Axioms of Design " Robot, Computer Integrated Manufacturing, Vol 6 no 2 133-142

[21] Suh, N.P , (1990), " The Principles of Design", Oxford University Press, New York.

[22] Kalyanmoy Deb, (1993). " Genetic Algorithms for Engineering Design Optimization." Proceedings of the advanced Study Institute on Computetional Methods for Engineering Analysis and design. Madras, 12 1-12.25.

[23] Goldberg, D. E, Dirk Thierens and K. Deb (1993). " Toward a better understanding of Mixing in genetic Algorithms." Journal of SICE, Vol 32(1).

[24] Schaffer, J. D (1984). " Some experiments in machine learnings using vector evaluated genetic algorithms." (TCGA file No. 00314) Ph D. dissertation, Vanderbilt University, Nashville, TN.

[25] Grefensttes, J. J. (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man and Cybernatics, SMC-16(1), 122-128

[26] Goldberg, D. E (1986). " Simple genetic algorithms and the minimal deceptive problems." (TCGA Report No. 86003).

[27] Fleming and Fonesca, P. J. (1993). " Genetic algorithms for multimodal optimization: formulation, discussion and generalisation. Proceedings of the Fifth International Conference on genetic Algorithms. San Mateo, CA

[28] Cavicchio, D. J. (1970). " Adaptive search using simulated evolution University of Michigen, Ann Arbor.

[29] Goldberg, D. E and Rechardson, J. (1987). " Genetic algorithms with sharing for Multimodal function optimization" Proceedings of the second international conference on Genetic algorithms

[30] Filippone, S. F. (1989). " Using Taguchi methods to apply to axioms of design". Robot, Computer Integrated Manufacturing. Vol·6 no.2: 133-142.

{31] Tribus and Szonyi G. (1989). " An alternative view of the Taguchi approach". Quality Progress, (May) Vol 22: 46-52.

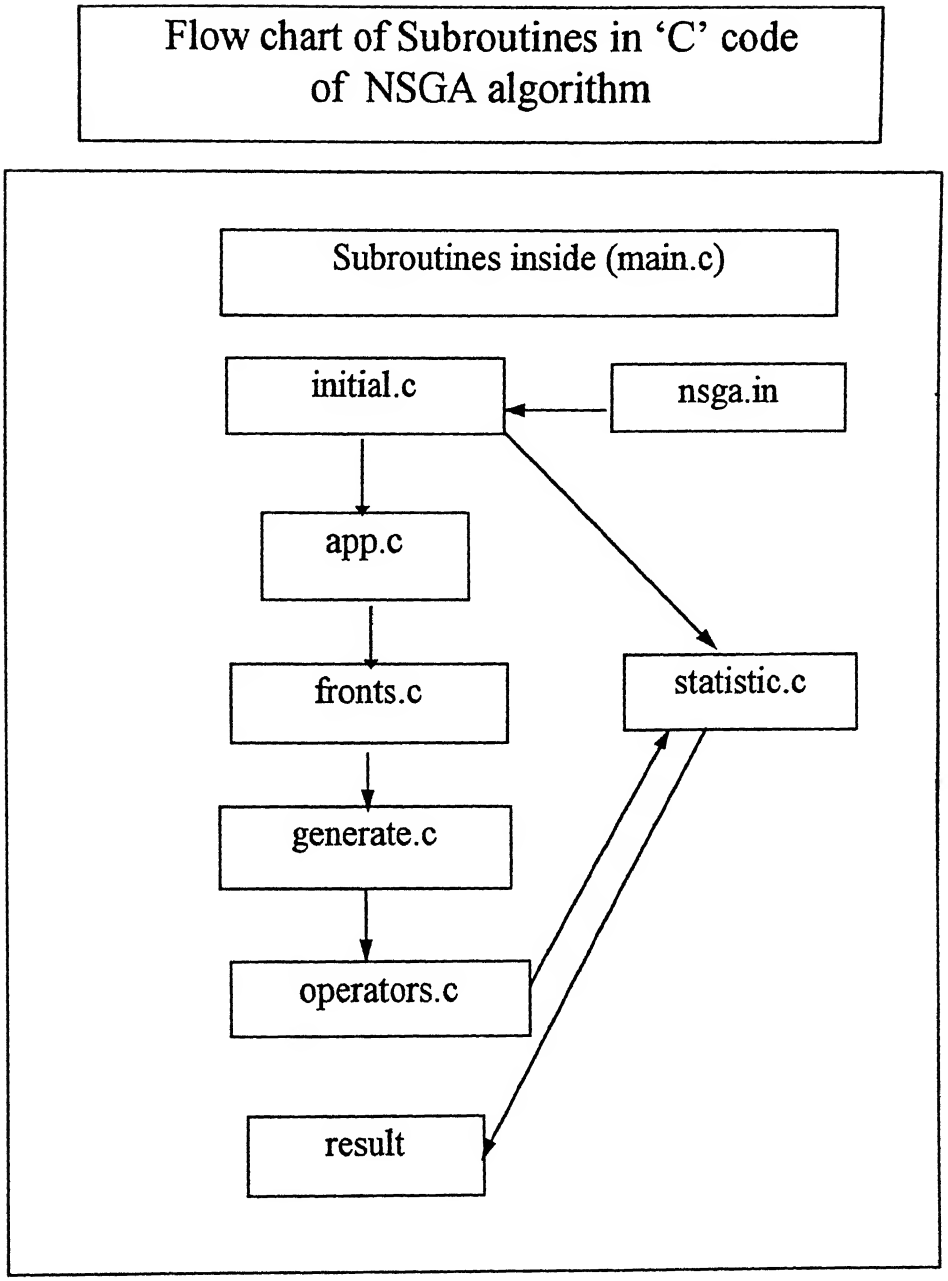
[32] Box , G. E P and Draper, N. R (1987)." Emprical model building and response surfaces " John wiley and sons N.Y.

[33] Amarendra Kumar (1996). " Use of Nondominated sorting in real coded GA." IIT Kanpur, M.Tech. Thesis, ME.

[34] Horn, Nafpliotis, N and Goldberg, D. E (1994). " The niched Pareto genetic algorithm for multi-objective optimization." Unpublished Manuscript.

[35] Way Kuo,- "Quality Through Engineering Design". Advances in Industrial Engineering Elsevier.

APPENDIX


Flow chart of Subroutines in 'C' code of NSGA algorithm
Subroutines inside (main.c)
initial.c
nsga.in
app.c
fronts.c
statistic.c
generate.c
operators.c
result